all rights reversed

HACKER

Αύγουστος 2015 # Τιμή τεύχους: 3.99 €

τεύχος 047 - Homebrew Mania Edition

Φτιάξτε τον δικό σας υπολογιστή, από το μηδέν [*]

Βελτίωση desktop application σε C#

Στρατηγικές εκμετάλλευσης name server

Ικανός caching name server, για το δίκτυό σας

Πώς τα Windows 10 μένουν μακριά από τα Windows 7

Προκαταρκτική σχεδίαση *αληθινά* προσωπικού υπολογιστή

[*] ...Καμία σχέση με συναρμολόγηση PC.

all rights reversed

HACKER

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Πέτρος Κυλαδίτης, Νίκος Μουρατίδης, Θωμάς Σερμπίνης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, @arkoudos
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.

Πλαστική ελευθερία

του @arkoudos

Τις τελευταίες μέρες παίζουν, ταυτόχρονα, τρεις-τέσσερεις ειδήσεις που μου κάνουν αίσθηση.

Είδηση 1. Οι δημόσιοι υπάλληλοι και οι συνταξιούχοι να κάνουν υποχρεωτικά χρήση πλαστικού χρήματος. Θα κάνουν ανάληψη 120-150 ευρώ την εβδομάδα, αναφέρεται σε δημοσίευμα, ενώ τα υπόλοιπα θα μπορούν να τα χρησιμοποιούν μόνο μέσω κάρτας. Παραθέτω απόσπασμα από το The TOC:

> *Συγκεκριμένα το ΥΠΟΙΚ επεξεργάζεται σχέδιο σύμφωνα με το οποίο θα προβλέπεται η υποχρεωτική χρήση καρτών στις συναλλαγές των συνταξιούχων και των δημοσίων υπαλλήλων.*
>
> *Αυτό θα επιτυγχάνεται, σύμφωνα με το δημοσίευμα, με έλεγχο στα μετρητά που σηκώνουν από μισθούς και συντάξεις και συγκεκριμένα περιορίζοντας την εβδομαδιαία ανάληψη μετρητών από τις τράπεζες στα 150 ευρώ από τα 420 που είναι σήμερα. Τα χρήματα που απομένουν στους λογαριασμούς των δικαιούχων, συνταξιούχων και δημοσίων υπαλλήλων, θα μπορούν να ξοδεύονται μόνο μέσω πιστωτικών ή χρεωστικών καρτών.*

Είδηση 2. Οι γιατροί υποχρεούνται να πληρώνονται με πλαστικό χρήμα. Οι γιατροί, με δεδομένες κάποιες προϋποθέσεις, έκαναν κατ' αρχάς δεκτό το μέτρο.

Είδηση 3. Όλες οι καινούργιες επιχειρήσεις από 1/1/2016 θα πληρώνονται με πλαστικό χρήμα. Αυτό θα επεκταθεί υποχρεωτικά και στις επιχειρήσεις που έχουν ξεκινήσει παλαιότερα, και σύντομα σε όλες. Παραθέτω απόσπασμα:

> *Όλες οι νέες επιχειρήσεις που θα συσταθούν το 2016 θα υποχρεούνται για όλες τις συναλλαγές τους να χρησιμοποιούν χρεωστικές ή πιστωτικές κάρτες. Αυτό σημαίνει ότι θα εγκαταστήσουν μηχάνημα POS το οποίο θα επιδοτείται από τα πρόγραμμα του ΕΣΠΑ. Οπως αναφέρει στην «Κ» ο αναπλ. υπουργός Οικονομικών Τρ. Αλεξιάδης, εφόσον οι επιχειρηματίες χρησιμοποιούν έναν λογαριασμό και μία κάρτα για τα έσοδα και τα έξοδά τους, θα απαλλάσσονται από την υποβολή της δήλωσης φόρου εισοδήματος και την περιοδική δήλωση ΦΠΑ, δεδομένου ότι όλα τα στοιχεία θα είναι γνωστά στο υπουργείο. Σταδιακά το μέτρο της μη υποβολής δήλωσης θα επεκταθεί και στις υπόλοιπες επιχειρήσεις.*

Είδηση 4. (Αυτή ήταν στα ψιλά.) Οι τράπεζες είναι υποχρεωμένες, μετά από απλό αίτημα μίας δημόσιας επιχείρησης (δήμοι, ΔΕΚΟ, εφορίες κ.ο.κ.) να παρακρατούν από λογαριασμούς πολιτών τυχόν χρέη τους. Επίσης, το όριο του προστατευόμενου ποσού κατέβηκε από τα 1500 ευρώ στα 1000. Παραθέτω απόσπασμα:

> *Έστω ότι έχετε απλήρωτη μία κλήση από δημοτική αστυνομία, ή οφείλετε Χ ποσό από νερό στον δήμο που διαμένετε, ή έχετε οφειλή από ΔΕΗ, τότε είναι πολύ πιθανό αν έχετε τραπεζική κατάθεση, να δείτε το ποσό οφειλής να αφαιρείται από τον τραπεζικό σας λογαριασμό.*

/var/log/messages

> *Συγκεκριμένα από το καλοκαίρι (η μεγάλη έξαρση σημειώθηκε το Σεπτέμβριο), οι τράπεζες παραλαμβάνουν ομαδικά κατασχετήρια –από εφορίες, τελωνεία, δήμους, ΔΕΚΟ, ασφαλιστικά ταμεία–, για δέσμευση τραπεζικών λογαριασμών.*
>
> *Τα αιτήματα είναι πολλά, μα πάρα πολλά. Υπολογίζεται ότι καθημερινά φτάνουν στις τράπεζες κατασχετήρια που αφορούν 500 και πάνω ονόματα.*
>
> *Με τα κατασχετήρια ζητείται η δέσμευση καταθέσεων και μάλιστα για ποσά που συνήθως δεν ξεπερνούν τα 150 ευρώ.*
>
> *Πρόκειται για οφειλές, από λογαριασμούς νερού, ρεύματος, ακόμα και από κλήσεις της δημοτικής αστυνομίας.*
>
> *Ο καταθέτης δεν έχει κανένα τρόπο αντίδρασης και άμυνας, αφού ο νόμος προβλέπει σαφώς την κατάσχεση οφειλής, από τραπεζικό λογαριασμό.*

Όμως εδώ οφείλω να παραθέσω και δύο προσωπικές περιπτώσεις –πιθανώς θα υπάρχουν κι άλλες–, με κατασχέσεις για ποσά κάτω από το προστατευόμενο όριο [*]. Στη δεύτερη περίπτωση μάλιστα η κατάσχεση έγινε γιατί η παθούσα ήταν ...εγγυήτρια.





Επαναλαμβάνω ότι αν και οι περιπτώσεις είναι μόνο δύο, σίγουρα θα υπάρχουν εκατοντάδες παρόμοιες.

[*] Προστατευόμενο όριο υπάρχει μόνο σε έναν λογαριασμό κατ' άτομο και για ποσό όχι μεγαλύτερο των 1000 ευρώ.

Εντάξει με την ενημέρωση; Πάμε στις σκέψεις τώρα.

Είναι πολύ δύσκολο να μεταφέρω με λέξεις αυτό που αισθάνομαι. Το να επιτρέψω να παραδοθεί το χρήμα στις τράπεζες, εγώ να έχω μόνον έναν αριθμό που θα πιστοποιεί ότι μου ανήκει, να είμαι διαρκώς ελεγχόμενος για το τι αγοράζω με τα χρήματα που

κέρδισα ο ίδιος, να μπορεί ο καθένας να ξέρει την οικονομική μου κατάσταση και να μην μπορώ να αντιδράσω ακόμη κι αν αδικούμαι, δημιουργούν συνθήκες που με εγκλωβίζουν αφόρητα.

Για τη συνέχεια κανονικά θα έπρεπε να γράψω "οι άνθρωποι" αλλά γράφω "εγώ", ώστε να μην υπάρχει καμία παρεξήγηση για το "χειρότερο σενάριο" που υπερασπίζομαι.

Αν για μένα έχει μεγαλύτερη σημασία να πληρωθεί το φαγητό του παιδιού μου, από το να πληρωθεί ένα χρέος στην τράπεζα με τα χρήματα που έχω κερδίσει, αυτό θα έπρεπε να είναι μια απόφαση ιερή. Αυτή η προτεραιότητα είναι δική μου ευθύνη. Τα χρήματα που κερδίζω είναι δικά μου, και η διαχείρισή τους βαρύνει αποκλειστικά εμένα.

Αν, σύντομα, καμία συναλλαγή δεν θα επιτρέπεται χωρίς πλαστικό χρήμα, μου φαίνεται πολύ ενοχλητικό το γεγονός ότι η τράπεζα, και τόσοι άλλοι πίσω απ' αυτή, θα έχουν την απόλυτη καταγραφή όλων των οικονομικών συνηθειών μου, των οικονομικών μου σχέσεων, της οικονομικής μου ταυτότητας.

Μου φαντάζει απολύτως αδιανόητο για την προσωπική μου ελευθερία, το ότι τα χρήματά μου θα είναι ιδιοκτησία άλλου. Να μου τα δίνει, όποτε κι όπως νομίζει, να τα διαχειρίζεται όπως νομίζει, να μου τα παρακρατεί όποτε νομίζει.

Μου φαίνεται πολύ ανησυχητικό ένα σενάριο κατά το οποίο, αύριο το πρωί, το κράτος αποφασίζει μια έκτακτη εισφορά για όλους, π.χ., της τάξης του 5%-10% επί των τραπεζικών καταθέσεων, και τα αντίστοιχα ποσά να μεταφέρονται από έναν λογαριασμό σε άλλον, χωρίς καμία δυνατότητα για αντίδραση.

Μου φαίνεται εξαιρετικά τρομακτικό να αποφασιστεί, π.χ., η επιστροφή σε εθνικό νόμισμα, και εν μία νυκτί για μας να αλλάξουν το εικονίδιο και βεβαίως οι αξίες που σχετίζονται με τις ηλεκτρονικές πληρωμές, χωρίς να έχουμε καμία δυνατότητα για την όποια αντίδραση.

Τα χρήματά μας δεν θα είναι πια δικά μας. (Η κατοχή τους ήταν μια αυταπάτη, το ξέρω – δεν ήταν ποτέ δικά μας.) Θα ανήκουν στο κράτος και στις τράπεζες και θα τα διαχειριζόμαστε μόνο με τη διακριτική τους ευχέρεια, αν κι όπως εκείνοι επιτρέπουν. Όπως λέω πάντα, προσοχή στο "αν δεν έχεις σκοπό να παρανομήσεις, δεν έχεις τίποτα να χάσεις". Αυτό είναι το μότο που προηγείται της παράδοσης των πλέον σημαντικών ελευθεριών σε άλλους.

Αυτό που για την παρούσα κυβέρνηση, όταν ήταν αντιπολίτευση, ήταν ο "οικονομικός στραγγαλισμός μιας ολόκληρης χώρας από τους δανειστές της", με τον οικονομικό έλεγχο που επιχειρείται να εφαρμοστεί αφορά πλέον στο ατομικό επίπεδο: Ο προσωπικός πλούτος περνά στα χέρια του κράτους και της εξουσίας.

Επιπλέον, οι "too big to fail" τράπεζες δεν θα έχουν μόνο τον έλεγχο μιας οικονομίας, αλλά και τον απόλυτο έλεγχο του χρήματος κάθε πολίτη ξεχωριστά. Κάνουμε ένα μεγάλο, επικίνδυνο θηρίο, απλώς μεγαλύτερο – κι αφάνταστα πιο επικίνδυνο.

Έχοντας βάλει ξεκάθαρα, εδώ και χρόνια, την οικονομία ως ισότιμο συνομιλητή των ηθικών μας αξιών, η μετατροπή της όποιας οικονομικής μας ελευθερίας σε πλαστική με ανησυχεί βαθιά. Προϊδεάζει δε ένα εξαιρετικά μαύρο μέλλον, κατά την ταπεινή μου γνώμη.



σνέλ μέλ

Αγαπητοί μου,

Καθυστέρησα να αρχίσω την επιστολή μου σήμερα γιατί ξεφυλλίζω, από το πρωί, ένα βιβλίο μαθηματικής λογικής που μου άφησε ο ανιψιός πριν φύγει για Λονδίνο.

Νόμιζα πως το είχε ξεχάσει. "Εδώ θα το αφήσεις;" τον ρώτησα την ώρα που έμπαινε στο ασανσέρ.

"Ναι", μου λέει. "Εις μνήμην".

Από το πρωί λοιπόν έχω βάλει ένα ούζο ("εις μνήμην") και το ξεφυλλίζω.

Ξανασκεφτόμουν την θεωρία του Τάρσκι περί αλήθειας, και το μυαλό μου γυρνάει και ξαναγυρνάει σε κάτι το οποίο, έτσι τουλάχιστον όπως το καταλαβαίνω τώρα, βλέπω πως είναι η βασική της εννοιολογική ανεπάρκεια.

ΟΚ, δεν ισχύει αυτό απολύτως. Δεν πρόκειται τόσο για πραγματική ανεπάρκεια. Καλύτερα θα ήταν να το χαρακτήριζα εννοιολογικό της περιορισμό. Αναπόφευκτο ίσως. (Αυτό πιστεύει ο Τάρσκι.)

Θα θυμάστε ίσως τι μας λέει. Τα λέγαμε και πέρσυ.

Ο Τάρσκι, ακολουθώντας το παράδειγμα των σχολαστικών φιλοσόφων του μεσαίωνα, εξηγεί πως η αλήθεια μιας πρότασης S σε μια (τυπική) γλώσσα L πιστοποιείται από την αντιστοιχία της πρότασης αυτής με μια συγκεκριμένη πραγματικότητα την οποία η πρόταση προσπαθεί να αποτυπώσει.

Αντιγράφω από την επιστολή #034. Εγραφα τότε:

" Κατά Τάρσκι μια πρόταση είναι αληθής αν (και μόνο αν) περιγράφει / αντικατοπτρίζει / αντιστοιχεί σε μια συγκεκριμένη πραγματικότητα. "The cat is on the mat" is true if and only if the cat is on the mat.

"Τσεκάρουμε: Είναι η γάτα στο χαλάκι; Είναι. Άρα μια πρόταση που μας λέει πως "η γάτα είναι στο χαλάκι", είναι αληθής.

"Τσεκάρουμε ξανά. Εφυγε η γάτα από το χαλάκι; Χάθηκε το χαλάκι; Δεν είχαμε ποτέ γάτα; Η γάτα μας δεν κάθεται ποτέ στο χαλάκι, γιατί δεν της αρέσει; Τότε το "η γάτα είναι στο χαλάκι" είναι ψευδής."

Ωραία όλα αυτά αλλά σα να λείπει κάτι. Ο Τάρσκι μας λέει πότε μια πρόταση είναι ή δεν είναι αληθής, αλλά δεν μας λέει πώς θεμελιώνεται η ίδια η έννοια της αλήθειας.

Αυτονόητο αυτό, θα μου πείτε. Πώς αλλιώς; Πώς αλλιώς θα μπορούσες να αποφανθείς για την αλήθεια ή μη μιας πρότασης αν δεν ξέρεις ήδη τι είναι αυτό που ονομάζεται αλήθεια; Θα ήταν σαν να προσπαθείς να αποφασίσεις αν κάτι είναι πράσινο ή κόκκινο χωρίς να έχεις επίγνωση της έννοιας του χρώματος.

Στον Τάρσκι, λοιπόν, η έννοια της αλήθειας μοιάζει σαν να προκύπτει κυκλικά. Είναι σαν να μας λέει πως "αλήθεια" είναι αυτό που έχει μια πρόταση όταν η πρόταση είναι "αληθής", αλλά χωρίς να μας λέει τι σημαίνει αυτό. Δεν μας δίνει, δηλαδή, ορισμό της αλήθειας.

Αυτό δεν τον απασχολεί. Μας το λέει άλλωστε ρητά: Ο ορισμός της αλήθειας σε μια γλώσσα L δεν μπορεί να δοθεί παρά μόνο έξω από την L, σε μια μεταγλώσσα μL της L. Δεν γίνεται αλλιώς. Τελεία και παύλα.

Το κακό είναι πως αν, όπως ο Τάρσκι, αποδεχτούμε ως αναπόφευκτο αυτό τον περιορισμό, τότε θα πρέπει να συμφιλιωθούμε με μια βασική του επίπτωση: Αν κάθε γλώσσα χρειάζεται τη μεταγλώσσα της, τότε κάθε μεταγλώσσα χρειάζεται τη μετα-μεταγλώσσα της. Και πάει λέγοντας.

Πού τελειώνει όλο αυτό; Πουθενά. Συνεχίζεται επ' αόριστον.

Σαν να λέμε πως "έχει χελώνες μέχρι κάτω", όπως έγραφε και στο εντιτόριαλ του τεύχους #006.

Ε, λοιπόν, εμένα αυτό δεν μου αρέσει καθόλου.

Όχι μόνο σε 'μένα, δηλαδή.

Κι εδώ ερχόμαστε στο βιβλίο που μου άφησε ο ανιψιός.

"The Principles of Mathematics Revisited", υπό Jaakko Hintikka.

Γιάακκο Χίντικκα: Μαθηματικός. Φιλόσοφος της λογικής. Σημαντικός. Εκ Φινλανδίας. Έφυγε στα μέσα Αυγούστου. Σε βαθύ γήρας.

Το βασικό εγχείρημα, λοιπόν, του Χίντικκα, είναι να παρακαμφθεί η ανωμαλία των πολλαπλών χελωνών, με την βοήθεια μιας λογικής πρώτου βαθμού, όπου έννοιες όπως αυτή της αλήθειας των προτάσεων S μιας γλώσσας L θεμελιώνονται, ορίζονται (και διατυπώνονται) στο ίδιο αυτό επίπεδο της γλώσσας L, χωρίς την ανάγκη να ανατρέχει κανείς σε κάποια μεταγλώσσα μL της L.

Πώς; Με το να περιγράψεις τη διαδικασία εύρεσης της τιμής αληθείας μιας πρότασης με όρους θεωρίας παιγνίων.

Σύμφωνα με τον Χίντικκα, το να αναλογίζεσαι αν μια πρόταση S (σε μια γλώσσα L) είναι αληθής ή όχι, είναι σαν να παίζεις ένα συγκεκριμένο παιχνίδι, ας το πούμε G, που έχει συγκεκριμένους κανόνες. Δεν έχει σημασία ποιοι είναι οι κανόνες. Σημασία έχει πως είναι γνωστοί και πως τους αποδέχεσαι. Τους ακολουθείς όσο πιο πιστά μπορείς, όντας υπό τη συνεχή επίβλεψη του αντιπάλου σου στο παιχνίδι, που προσπαθεί κι αυτός, με τους ίδιους κανόνες, να σε νικήσει.

Το ερώτημα για το αν μια πρόταση είναι αληθής ή όχι μπορεί να αναχθεί στο ερώτημα αν υπάρχει νικηφόρα στρατηγική για τον ένα από τους δύο "παίκτες". Δεν χρειάζεται να ανατρέξει κανείς σε κάποια μεταγλώσσα. Οι κανόνες του παιχνιδιού και η νικηφόρα στρατηγική (αν υπάρχει) διατυπώνονται στην ίδια ακριβώς γλώσσα, στην οποία είναι διατυπωμένη και η υπό εξέτασιν πρόταση.

Πραγματικά εντυπωσιακό, κυρίως λόγω της απλότητας της ιδέας.

Σκεφτείτε το. Ο παίκτης Α θέλει να επιβεβαιώσει την πρόταση και ο παίκτης Β να τη διαψεύσει. Και οι δύο παίζουν με τους ίδιους κανόνες. Λέει λοιπόν, ο Χίντικκα: Αν υπάρχει κατάλληλη στρατηγική στο παιχνίδι



G που επιτρέπει στον παίκτη A να επιβεβαιώσει την πρόταση S, τότε η πρόταση S μπορεί να οριστεί ως "αληθής". Αν υπάρχει κατάλληλη στρατηγική που επιτρέπει στον παίκτη B να απορρίψει την πρόταση S, τότε η πρόταση S μπορεί να οριστεί ως "ψευδής". Το όλο ερώτημα περί αλήθειας ανάγεται στο ερώτημα αν υπάρχει νικηφόρα στρατηγική είτε για τον A είτε για τον B.

Η ιδέα του Χίντικκα, την οποία απλουστεύω και σχηματοποιώ όσο δεν παίρνει, ακούγεται ελαφρώς εκκεντρική αλλά δεν είναι και τόσο. Ακόμα κι όταν έχεις να αποδείξεις μια πρόταση με τον συμβατικό τρόπο, αυτό που κάνεις είναι να ακολουθείς κατάλληλα εκείνα τα συλλογιστικά βήματα που σου επιτρέπουν είτε να ολοκληρώσεις την απόδειξη είτε να δείξεις ότι δεν γίνεται. Όταν λες ότι υπάρχουν τέτοια συλλογιστικά βήματα είναι σαν να λες, κατά Χίντικκα, ότι υπάρχει κατάλληλη στρατηγική που θα σου επιτρέψει να νικήσεις στο παιχνίδι G.

Η αλήθεια δεν ορίζεται σε αναφορά με τη σχέση που έχουν οι προτάσεις μιας γλώσσας L με έναν κόσμο (όπως στον Τάρσκι), αλλά σε αναφορά με την ύπαρξη (ή όχι) νικηφόρων στρατηγικών σε παίγνια G εντός των ορίων της γλώσσας L, στην οποία διατυπώνονται οι προτάσεις αυτές.

Η δύναμη, για να το πω αλλιώς, της θεωρίας, είναι το ότι κατασκευάζει μια πρωτοβάθμια λογική η οποία επαρκεί για τη διατύπωση και διερεύνηση ερωτημάτων που παραδοσιακώς θα προσεγγίζονταν με τη βοήθεια επιπέδων και μεταεπιπέδων. Αποφεύγει, δηλαδή, τις "χελώνες μέχρι κάτω", αλλά με τρόπο λογικώς αυστηρό, εσωτερικώς συνεκτικό και χωρίς να υποπίπτει στην παγίδα του σχετικισμού.

Κι αυτό ισχύει τόσο στις τυπικές γλώσσες (π.χ., στα μαθηματικά), όσο και στις φυσικές. Η αλήθεια θεμελιώνεται στις ανθρώπινες δραστηριότητες μέσω των οποίων αποκτά νόημα, και δεν μας επιβάλλεται με το ζόρι απ' έξω.

Για να το πούμε πιο απλά, με ένα παράδειγμα από τη ζωή βγαλμένο: Πάω στην ΕΒΓΑ του κυρίου Κώστα, όπου όλοι τσακώνονται με όλους. Φωνάζουν και ωρύονται για το αν υπαναχώρησε ο ΣΥΡΙΖΑ από τις αρχικές του υποσχέσεις και για το αν έπρεπε να γίνουν εκλογές ή όχι. "Συζητάνε". Μου ζητάνε να πάρω θέση.

Μεγάλη παγίδα, και δεν είμαι κορόιδο για να πέσω. Φεύγω γρήγορα χωρίς να απαντήσω. Αλλά ποια ακριβώς είναι η παγίδα;

Αν ερχόταν μαζί μου ο Χίντικκα στην ΕΒΓΑ του κυρίου Κώστα και του ζητούσαν να πάρει θέση, το πρώτο πράμα που θα έκανε θα ήταν να αναρωτηθεί: Υπάρχει νικηφόρα στρατηγική για τον "παίκτη" εκείνον που επιχειρεί να υποστηρίξει τη μία ή την άλλη άποψη;

Θα απαντούσε αμέσως. Όχι, δεν υπάρχει.

Κι αυτό γιατί στην ΕΒΓΑ του κυρίου Κώστα δεν ισχύουν δύο από τις βασικές προϋποθέσεις για την διαμόρφωση νικηφόρας στρατηγικής στο παιχνίδι G: Αφενός δεν έχουμε όλοι συμφωνήσει στο ποιο είναι το πεδίο αναφοράς της γλώσσας L στην οποία "συζητάμε"· και αφετέρου, δεν έχουμε όλοι συμφωνήσει να δεσμευόμαστε με τον ίδιο τρόπο από τους κανόνες του παιχνιδιού G.

Η θεωρία του Χίντικκα μας επιτρέπει να δούμε καθαρά αυτό που ήδη ξέραμε. Η συζήτηση στην ΕΒΓΑ του κυρίου Κώστα --το παίγνιο που λαμβάνει χώρα εκεί-- δεν έχει τόσο να κάνει με την εύρεση της αλήθειας αλλά με κάτι άλλο πολύ διαφορετικό.

Αυτά λοιπόν. Τώρα σας αφήνω.

Βάζω ένα ακόμα ουζάκι εις μνήμην, και επιστρέφω στο βιβλίο.

Σας ασπάζομαι,
Θείος Ακάκιος



Αυτό το Mac mini είναι σχεδιασμένο ώστε να κάνει ό,τι θέλετε – και πολλά περισσότερα.

Αυτό το Mac mini δεν είναι το βασικό μοντέλο, και **κοστίζει 1249 ευρώ** (http://store.istorm.gr/mac/mac-mini/2-8ghz-mac-mini.html).

Αυτό το Mac mini το **κληρώνουμε μεταξύ 25 αναγνωστών** με Advanced Access.

Αν η συνδρομή σας πλησιάζει στη λήξη της, αυτό το Mac mini **είναι εξαιρετικά πιθανό** να γίνει δικό σας.

Για να πάρετε μέρος στην κλήρωση, **αρκεί να αποκτήσετε Advanced Access** συμπληρώνοντας τη φόρμα παραγγελίας στο http://deltahacker.gr/pdf12order/#s2.

Από την ίδια τη φόρμα παραγγελίας (http://deltahacker.gr/pdf12order/#s2) βλέπετε **πόσους αναγνώστες με Advanced Access χρειαζόμαστε ακόμα**, ώστε να προχωρήσουμε στην κλήρωση.

Καλή σας επιτυχία!

Skill: Beginner
Tags: homebrew computer, ATmega644

Κατασκευάστε έναν υπολογιστή από το μηδέν, μέρος 1

Πιο προσωπικός, δεν γίνεται!

Το πιο όμορφο "Hello World" δεν γράφεται σε κάποια υπερσύγχρονη γλώσσα αλλά στην ταπεινή BASIC, η οποία είναι τόσο παλιά ώστε πολλοί αναγνώστες ενδέχεται να μην την συνάντησαν ποτέ. Θέλουμε επιπρόσθετα να πιστεύουμε ότι το δικό μας "γεια σου κόσμε" είναι πανέμορφο, όχι γιατί είναι περίτεχνα γραμμένο αλλά γιατί τρέχει σ' έναν υπολογιστή που κατασκευάσαμε μόνοι μας. Κι όταν λέμε "κατασκευάσαμε", δεν εννοούμε φυσικά τη συναρμολόγηση ενός PC.

του Παναγιώτη Βαρελά

Παίζει παράξενα παιχνίδια ο νους του ανθρώπου. Μια αλλόκοτη ιδέα, όχι κατ' ανάγκη πρωτότυπη αλλά σίγουρα ενδιαφέρουσα, είναι πιθανό να ξεφυτρώσει από τη σκέψη μας ανά πάσα στιγμή. Είναι τότε που διάφορες ασύνδετες πληροφορίες ξεπηδούν από άσχετες γωνιές του μυαλού κι εντελώς ξαφνικά μοιάζουν να ταιριάζουν μεταξύ τους σαν τα κομμάτια ενός παζλ. Κάτι τέτοιο συνέβη και σε μένα, πριν από περίπου ένα χρόνο, καθώς προσπαθούσα να φρεσκάρω τις γνώσεις μου στα CPLD. (Τα CPLD, από το Complex Programmable Logic Device, είναι προγραμματιζόμενα ολοκληρωμένα που ενσωματώνουν ένα μεγάλος πλήθος λογικών πυλών. Ο χρήστης μπορεί να συνθέτει λογικά κυκλώματα, συνδυάζοντας τις διαθέσιμες πύλες κατά το δοκούν.) Όλα τα σχετικά βιβλία ξεκινούσαν με τη μελέτη και την υλοποίηση μικρών λογικών κυκλωμάτων, εν είδει "Hello World". Η μεταφορά αυτής της προγραμματιστικής άσκησης στη σχεδίαση του hardware μού θύμισε πόσο κοντά βρίσκεται ο κόσμος του λογισμικού με τον κόσμο του υλικού, όταν ασχολείται κανείς με μικροελεγκτές ή με CPLD και FPGA. Κάπως έτσι, η σκέψη μου έτρεξε σε μια σειρά άρθρων για την παραγωγή ήχου και τη μετατροπή του Arduino σ' ένα απλοϊκό synthesizer. Στη συνέχεια θυμήθηκα δύο εντελώς διαφορετικά άρθρα που αφορούσαν στην παραγωγή σήματος VGA με τη βοήθεια ενός μικροελεγκτή, κι από 'κει κατέληξα να σκέφτομαι ένα άρθρο του Γιώργου (aka @giannoug) στο τεύχος 014 (http://deltahacker.gr/deltahacker014), που περιέγραφε την "εγκατάσταση" της tiny BASIC στο Arduino (http://deltahacker.gr/?p=7035). Ε, λοιπόν, όλα αυτά τα άσχετα θέματα συνδυάστηκαν άψογα και με οδήγησαν σε μια συναρπαστική ιδέα: Θα μπορούσα να κατασκευάσω έναν υπολογιστή από το μηδέν και, εκεί, στο δικό του περιβάλλον και στη δική του γλώσσα, θα μπορούσα να γράψω ένα πραγματικά εντυπωσιακό "Hello World". Ουσιαστικά θα έγραφα ένα τυπικό "Hello World", μόνο που προηγουμένως θα έφτιαχνα τον υπολογιστή ο οποίος θα έτρεχε το αντίστοιχο πρόγραμμα.



Η μνήμη RAM του υπολογιστή μας είναι απειροελάχιστη, ακόμα και για τα δεδομένα των υπολογιστών της δεκαετίας του '80. Το ίδιο ισχύει και για τη μνήμη ROM, αλλά εμείς δεν έχουμε κανένα παράπονο. Η αποστολή αυτού του χειροποίητου υπολογιστή δεν ήταν να σπάσει κάποιο ρεκόρ σε τεχνικά χαρακτηριστικά, ούτε σε επιδόσεις.

Πραγματικό Personal Computer

Περιττό να σας πω ότι αυτή η ιδέα κατέλαβε πλήρως τη σκέψη μου κι από τη σύλληψή της μέχρι σήμερα, έχασα κάθε διάθεση να ασχοληθώ με τα CPLD. Για το όλο εγχείρημα στράφηκα και πάλι στους 8μπιτους μικροελεγκτές της οικογένειας AVR, με τους οποίους είμαι βαθιά εξοικειωμένος. Η προτίμηση στο "δοκιμασμένο υλικό" δεν πρέπει να ερμηνευτεί σαν υποχώρηση. Είχα αποφασίσει να φτιάξω έναν κανονικό υπολογιστή και δεν σκόπευα να κάνω εκπτώσεις. Αυτό βέβαια δεν σημαίνει ότι είχα και αυταπάτες. Η χρήση των 8μπιτων μικροελεγκτών έθετε χαμηλό ταβάνι για τις επιδόσεις. Σε γενικές γραμμές, ο υπολογιστής μου θα βρισκόταν στα ίδια επίπεδα με εκείνους της δεκαετίας του '80. Μιλάμε για την εποχή που η IBM έκανε διάσημο τον όρο PC. Η σκέψη ότι σχεδίαζα έναν τέτοιο υπολογιστή από το μηδέν, μου έδινε την αίσθηση ότι ορίζω εκ νέου την έννοια του Personal Computer. Ναι, υπερβάλλω λίγο, αλλά αυτό δεν σημαίνει ότι δεν βρήκα τη σκέψη διασκεδαστική.



Μετά από τόση δουλειά, για να φτιάξω τον υπολογιστή στον οποίο θα έγραφα το "hello world", αποφάσισα να το ρίξω έξω. Το πρόγραμμά μου τυπώνει το κλασικό μήνυμα, χρησιμοποιώντας το αγαπημένο μου χρώμα.

Προσωπικοί υπολογιστές

Τα πρώτα Personal Computers εμφανίστηκαν στα μέσα της δεκαετίας του '60. Ο χαρακτηρισμός "personal" δεν υπονοούσε ότι το σύστημα είχε χαμηλές επιδόσεις ή ότι προοριζόταν για απλές δουλειές. Η NASA είχε αγοράσει αρκετά κομμάτια από ένα μηχάνημα που θεωρείται ως το πρώτο PC – το Programma 101, από την Ιταλική Olivetti. Σκεφτείτε ότι οι συγκεκριμένοι υπολογιστές αξιοποιήθηκαν για τους υπολογισμούς του προγράμματος Apollo 11, που είχε σαν στόχο την προσεδάφιση στη Σελήνη. Η χρήση του όρου "προσωπικός" υπογράμμιζε την αντίθεση με τα Minicomputers και τα Mainframes, που απαρτίζονταν από ένα κεντρικό τμήμα και μερικά τερματικά. Οι χρήστες αυτών των υπολογιστών εργάζονταν στα τερματικά, αλλά η λειτουργία του υπολογιστή προϋπέθετε την παρουσία εξειδικευμένου προσωπικού στην κεντρική μονάδα. Το 1981 η IBM κυκλοφόρησε έναν υπολογιστή που στηριζόταν στον επεξεργαστή Intel 8088 κι έτρεχε στα 4.77MHz. Το πλήρες όνομα του μηχανήματος ήταν "IBM Personal Computer model 5150", και

η τεράστια εμπορική επιτυχία του καθιέρωσε την ονομασία "IBM PC". Τα επόμενα χρόνια εμφανίστηκαν υπολογιστές που στηρίζονταν σε κλώνους/αντίγραφα του 8088 κι έτσι προέκυψαν τα περίφημα "IBM PC compatible". Αυτή η ονομασία απλοποιήθηκε σύντομα σε ένα σκέτο "PC".

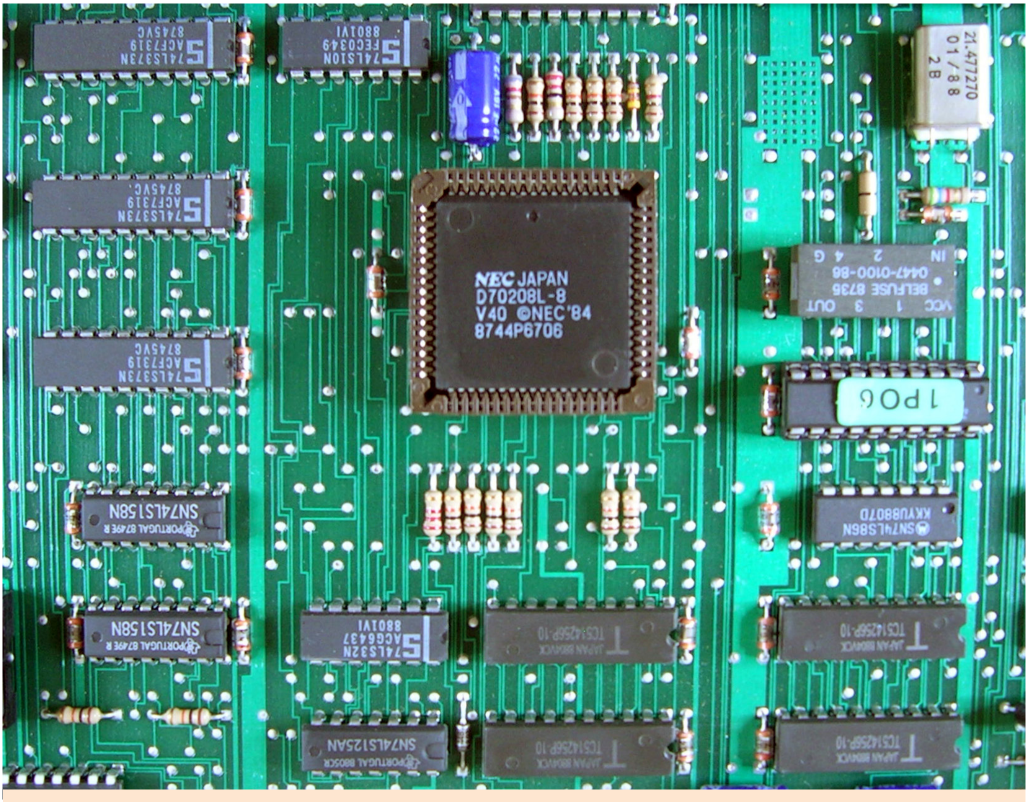


Αυτό το πρόγραμμα επιδεικνύει τη μισή χρωματική παλέτα του συστήματος. Θα μπορούσε άνετα να επιδεικνύει το σύνολο των χρωμάτων, αλλά δεν θα χωρούσαν στην οθόνη *μαζί* με το πρόγραμμα ;)

Η εργασία μου ξεκίνησε απαντώντας σ' ένα βασικό ερώτημα: Τι χρειάζεται για να χαρακτηριστεί ένα ηλεκτρονικό σύστημα ως υπολογιστής; Η απάντηση που έδωσα ικανοποιούσε τις προσωπικές μου προσδοκίες και ήταν επηρεασμένη από διάφορα αγαπημένα μηχανήματα που μεσουρανούσαν 30 χρόνια πριν. Το σύστημά θα μου θα έπρεπε να ενσωματώνει τα ακόλουθα:

- Είσοδο για πληκτρολόγιο
- Γλώσσα προγραμματισμού
- Έξοδο εικόνας
- Έξοδο ήχου

Αν πιστεύετε ότι η γλώσσα προγραμματισμού δεν έχει θέση στις προδιαγραφές του hardware, κάνετε λάθος. Σχεδίαζα έναν υπολογιστή παρόμοιο μ' εκείνους που κυκλοφορούσαν τη δεκαετία του '80. Αυτό μπορεί να μη σας λέει τίποτα, αλλά πρέπει να έχετε υπόψη ότι πολλά μηχανήματα της εποχής ενσωμάτωναν τον interpreter κάποιας παραλλαγής της BASIC, στο BIOS. Απ' αυτή την άποψη, η γλώσσα προγραμματισμού ήταν πρακτικά αδύνατο να διαχωριστεί από το υλικό.

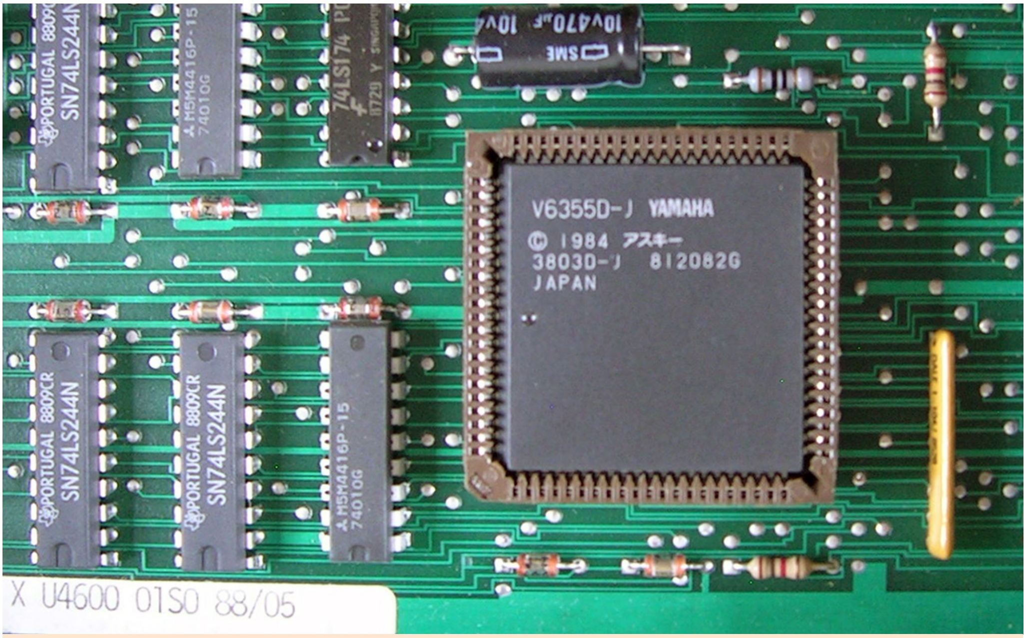


Η φωτογραφία δεν προέρχεται από κάποιο παλιό PC, αλλά από ένα "IBM PC compatible". Ο επεξεργαστής V40 της NEC αποτελεί κλώνο του 8088 της Intel. Πάντως ο συγκεκριμένος επεξεργαστής ενσωματώνει αρκετές βελτιώσεις, ενώ διαθέτει και περισσότερες εντολές σε σχέση με εκείνον της Intel.

Όπως αντιλαμβάνεστε, η παραπάνω λίστα δεν διατήρησε αυτή τη λακωνική μορφή για πολύ. Εξετάζοντας κάθε στοιχείο ως ξεχωριστό υποσύστημα, σκεφτόμουν και σημείωνα τα χαρακτηριστικά που θα ήθελα να έχει. Σε πολλές περιπτώσεις, εξάλλου, σημείωνα και λεπτομέρειες της υλοποίησης, που θα θεωρούσα ότι αργότερα θα μου φανούν χρήσιμες. Έτσι, σε σύντομο χρονικό διάστημα είχε προκύψει μια μακροσκελής λίστα τεχνικών προδιαγραφών, η οποία αποτέλεσε τον οδηγό για την υλοποίηση του υπολογιστή. Για να πάρετε μια γεύση, δείτε μερικά από τα χαρακτηριστικά του τελικού συστήματος:

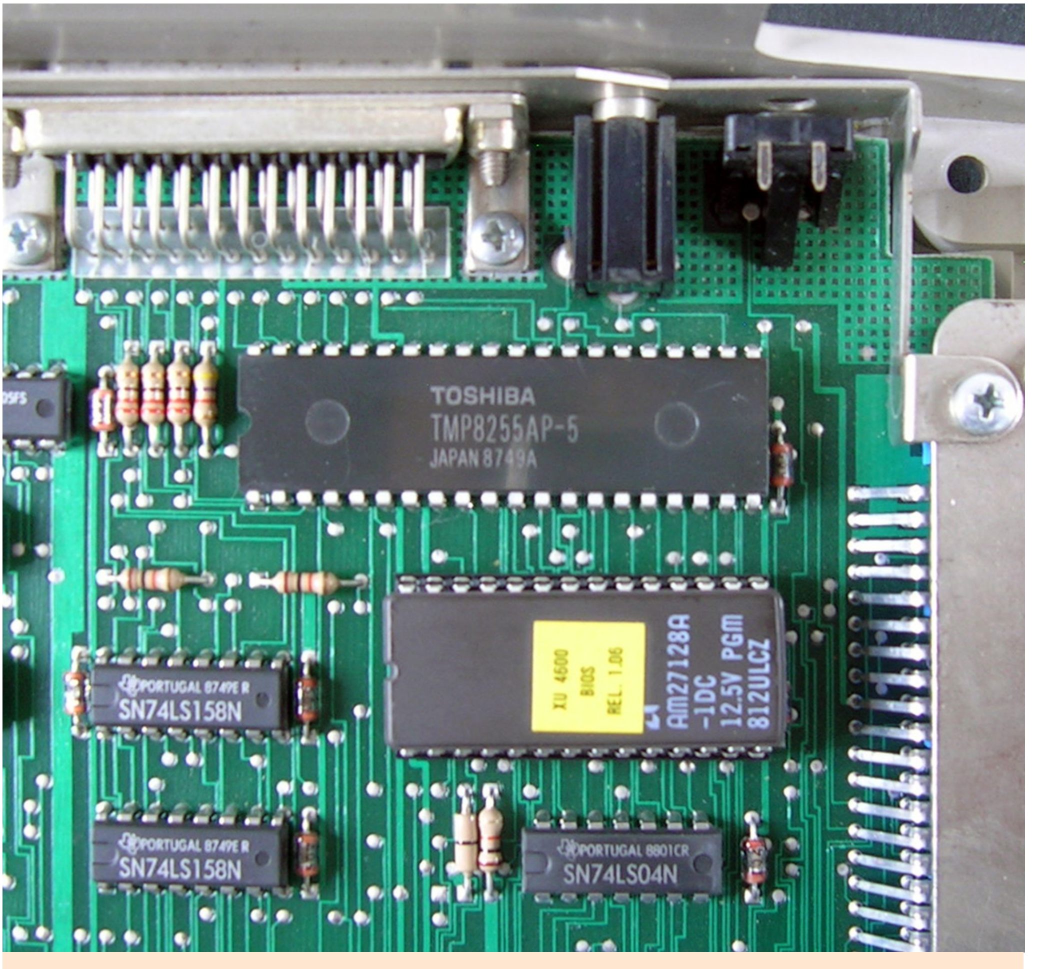
- Ανάλυση εικόνας: 256 x 240 pixels
- Κείμενο: 32 στήλες x 24 γραμμές
- Βάθος χρώματος: 8 bits (256 χρώματα)
- Ήχος: 4 ανεξάρτητα κανάλια
- Ακροδέκτες GPIO: 8 ψηφιακές είσοδοι/έξοδοι (λειτουργούν κι ως αναλογικές είσοδοι)
- BASIC με πρόσθετες εντολές για το χειρισμό του hardware
- Δυνατότητα αποθήκευσης και φόρτωσης προγραμμάτων
- Είσοδο για πληκτρολόγιο PS2

Η κατασκευή του υπολογιστή διήρκεσε λίγο παραπάνω από έναν χρόνο και ήταν γεμάτη με αμέτρητες σπαζοκεφαλιές. Μπορώ να θυμηθώ αρκετά στάδια της ανάπτυξης κατά τα οποία ήμουν έτοιμος να παραιτηθώ. Σκεφτείτε την απελπισία που ένιωθα όταν, για παράδειγμα, το firmware του "κεντρικού επεξεργαστή" παρουσίαζε κάποιο άγνωστο πρόβλημα και η επικοινωνία με το κύκλωμα γραφικών αποτύγχανε. Χωρίς έξοδο στην οθόνη και χωρίς παλμογράφο, έπρεπε να κάνω debug στα τυφλά.



Ένας μακρινός πρόγονος των σύγχρονων GPU. Μεταξύ άλλων, το datasheet αναφέρει ότι μπορεί να οδηγήσει οθόνες LCD και CRT, υποστηρίζει αρκετές διαφορετικές αναλύσεις, διαθέτει μνήμη που μπορεί να αξιοποιηθεί ως video RAM και υποστηρίζει τη σύνδεση light pen. Το πιο εντυπωσιακό όμως είναι η ικανότητα διαχείρισης του δρομέα. Ναι, η σχεδίαση του δρομέα αποτελεί εργασία για τη GPU.

Στην παρούσα σειρά άρθρων θα μελετήσουμε τη δομή και τη λειτουργία του υπολογιστή σε όλα τα επίπεδα. Θα εξετάσουμε τις σχεδιαστικές επιλογές που έγιναν στο hardware, τους στόχους και τους περιορισμούς, όπως επίσης και τα λάθη. Σε αντίθεση με την ενστικτώδη απάντηση που θα έδινε κανείς, σε αυτή την κατασκευή το hardware είναι πολύ πιο απλό από το software. Ο κατάλληλος προγραμματισμός των μικροελεγκτών αποτέλεσε μακράν τη μεγαλύτερη πρόκληση. Ακριβώς γι' αυτό, η παρουσίαση θα περιλαμβάνει και τη διεξοδική μελέτη του κώδικα, κάθε υποσυστήματος. Στο τέλος της διαδρομής θα μπορείτε να κατασκευάσετε κι εσείς τον συγκεκριμένο υπολογιστή ή, ακόμα καλύτερα, θα είστε σε θέση να σχεδιάσετε και να υλοποιήσετε τον δικό σας.



Το τσιπάκι με το κίτρινο αυτοκόλλητο προέρχεται από την AMD και είναι μια OTPROM (One Time Programmable ROM) των 16kB. Το τσιπ από την Toshiba αποτελεί έναν port extender και προσφέρει στον επεξεργαστή μερικούς πρόσθετους ακροδέκτες GPIO. Μόνο το Arduino νομίζατε ότι έχει αυτό το πρόβλημα;

Απαραίτητος εξοπλισμός

Για να παρακολουθήσει κανείς τη συγκεκριμένη σειρά άρθρων απαιτούνται βασικές γνώσεις στα ηλεκτρονικά και στον προγραμματισμό. Μην φαντάζεστε τίποτα ιδιαίτερο. Αν σας αρέσει η μελέτη και η κατασκευή ηλεκτρονικών κυκλωμάτων, οσοδήποτε απλά κι αν ήταν αυτά που έχετε συναντήσει ως τώρα, το υλικό του υπολογιστή δεν θα σας δυσκολέψει. Αντιστοίχως, αν έχετε προγραμματίσει οποιονδήποτε μικροελεγκτή, θα κατανοήσετε εύκολα τη γενική λογική του κώδικα που εμψυχώνει τον υπολογιστή.

Πέρα από τα παραπάνω εφόδια, για να προχωρήσει κανείς στην υλοποίηση της κατασκευής θα χρειαστεί κι ορισμένα εργαλεία. Τα απολύτως απαραίτητα σύνεργα είναι μόλις τρία:

- Ένα κολλητήρι με σχετικά λεπτή "μύτη"
- Ένα πολύμετρο με "buzzer συνέχειας" (continuity tester)
- Έναν hardware programmer για AVR

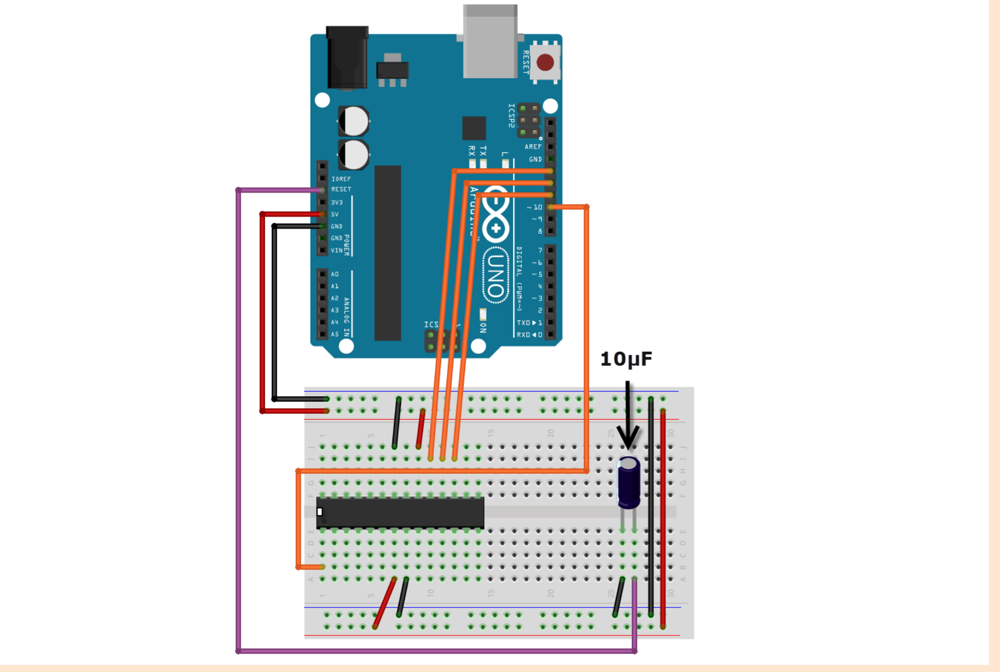
Για το κολλητήρι δεν υπάρχει κανένας ιδιαίτερος περιορισμός. Έχετε υπόψη ότι οι κολλήσεις είναι εύκολες, αλλά είναι και πάρα πολλές. Οι απαιτήσεις για το πολύμετρο είναι επίσης μηδαμινές. Το "buzzer συνέχειας" (μερικοί ηλεκτρολόγοι το αποκαλούν "τζιτζίκι") θα σας βοηθήσει να εντοπίσετε τις προβληματικές κολλήσεις, καθώς και τα πιθανά βραχυκυκλώματα. Όπως είπαμε ήδη, οι κολλήσεις είναι πάρα πολλές και είναι βέβαιο ότι θα γίνουν λάθη. Εγώ πάντως έκανα πάρα πολλά.



Ο μικρός hardware programmer της Atmel. Υποστηρίζεται από το Atmel Studio, από το Arduino IDE κι από κάθε γνωστό εργαλείο για τον προγραμματισμό των AVR. Αποτελεί άριστη επιλογή για όσους ξεκινούν τώρα και σκοπεύουν να ασχοληθούν σοβαρά.

Ο hardware programmer αποτελεί τη συσκευή που μεσολαβεί για τη μεταφορά ενός προγράμματος από τον υπολογιστή προς τον μικροελεγκτή. Μη σας ξεγελούν οι αυτοματισμοί που ενσωματώνει το Arduino. Ο προγραμματισμός ενός οποιουδήποτε μικροελεγκτή προϋποθέτει την παρουσία ενός hardware programmer. Εγώ αξιοποίησα ένα STK500 από την Atmel, μόνο και μόνο επειδή το είχα ήδη. Εσείς δεν είναι υποχρεωτικό να χρησιμοποιήσετε κάτι τόσο ακριβό. Μπορείτε να στραφείτε στον "entry-level" προγραμματιστή του ίδιου κατασκευαστή, που ονομάζεται AVRISP mkII και κοστίζει περί τα 30€. Αν αυτό το κόστος είναι απαγορευτικό τη δεδομένη περίοδο, μπορείτε να φτιάξετε το δικό σας programmer χρησιμοποιώντας ένα Arduino. Στην περίπτωση που επιλέξετε τη δεύτερη λύση, θα βρείτε αναλυτικές οδηγίες στο άρθρο "Ξεφύγετε από το Arduino" (http://deltahacker.gr/?p=9978) του τεύχους 025 (http://deltahacker.gr/deltahacker025).

Για την ώρα μπορείτε να αναπολήσετε τα μηχανήματα της εποχής του '80 κι αν δεν ανήκετε σε αυτούς που τα πρόλαβαν, μπορείτε να φανταστείτε τα χαρακτηριστικά που θα δίνατε σε έναν δικό σας υπολογιστή, αν τον χτίζατε από το μηδέν. Όταν η φαντασία καταντήσει ανιαρή γυρίστε στη σελίδα 62 για να διαβάσετε τη συνέχεια, όπου ξεκινάμε τη μελέτη του hardware.



Αν δεν έχετε και δεν προτίθεστε να αγοράσετε κάποιον hardware programmer, μπορείτε να επιστρατεύσετε το Arduino. Αρκεί να του φορτώσετε το πρόγραμμα "ArduinoISP" και να πραγματοποιήστε μια απλή συνδεσμολογία. Περισσότερες πληροφορίες θα βρείτε στο τεύχος 025, στο άρθρο "Ξεφύγετε από το Arduino".


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

Skill: Intermediate
Tags: recursion, caching, forwarding, BIND

Πώς δουλεύουν τα δίκτυα

Ο δικός μας caching name server

Θέλουμε να πιστεύουμε ότι τα τρία άρθρα περί DNS, που δημοσιεύτηκαν στο τεύχος 046, έχουν βάλει στο νόημα ακόμη κι εκείνους που είχαν μόνο μια αμυδρή ιδέα περί name servers και Domain Name System γενικότερα. Πράγματι, καλύψαμε σημαντικό έδαφος και πλέον έχουμε κατανοήσει επαρκώς τη λογική του όλου συστήματος. Τόσο πολύ που έχει έρθει η ώρα να φτιάξουμε τον δικό μας name server, για τις ανάγκες του δικού μας τοπικού δικτύου.

του Χρήστου Βαρελά

Σε πολύ λίγο, σε ένα σύστημα με Ubuntu Server θα εγκαταστήσουμε το δημοφιλές BIND (Berkeley Internet Name Domain, https://www.isc.org/downloads/bind) και θα το ρυθμίσουμε ώστε να λειτουργεί ως caching name server για τα μηχανήματα του τοπικού δικτύου. Το BIND θα συμπεριφέρεται είτε ως recursive είτε ως forwarding name server. Πρόκειται για απόφαση που θα πάρουμε σταθμίζοντας τις ικανότητες του (εικονικού ή φυσικού) μηχανήματος με το BIND, το διαθέσιμο bandwidth για τις εξερχόμενες συνδέσεις προς το Internet, όπως επίσης και τη στάση μας ως προς την εξάρτηση από άλλους name servers.

Ας συζητήσουμε τα προηγούμενα λίγο πιο αναλυτικά. Κατ' αρχάς η επιλογή του BIND δεν είναι μονόδρομος. Θα μπορούσαμε να επιλέξουμε άλλο λογισμικό, όπως, π.χ., είναι το Dnsmasq (caching και forwarding) ή το Unbound (caching και recursion ή forwarding). Όμως το BIND είναι εξαιρετικά ευέλικτο, κατάλληλο για απλά καθώς και για σύνθετα σενάρια, ενώ συναντάται συχνά στο Internet αλλά και σε τοπικά δίκτυα όλων των μεγεθών. Σκεφτήκαμε λοιπόν ότι, αν μη τι άλλο για εκπαιδευτικούς λόγους, δεν θα ήταν καθόλου κακή ιδέα να εξοικειωθούμε με το BIND, ξεκινώντας από τα ρηχά.

Σε ένα τυπικό οικιακό δίκτυο θεωρούμε ότι δεν υπάρχει θέμα με την ισχύ του μηχανήματος που θ' αναλάβει χρέη name server. Πράγματι, το ρόλο αυτό θα μπορούσε να έχει ακόμη κι ένα Raspberry Pi. Στην απόφασή μας, λοιπόν, για το αν ο name server θα είναι recursive ή forwarding, θα βαρύνουν άλλοι παράγοντες. Σημειώστε πάντως ότι για τους clients δεν έχει καμία σημασία τι από τα δύο θα είναι: Το μόνο που μετράει γι' αυτούς είναι η ικανότητα για caching – κι αυτή θα την έχουμε και στις δύο περιπτώσεις. Πώς πρέπει να συμπεριφέρεται ο name server μας, λοιπόν;

Recursive. Σ' αυτή την περίπτωση ο server δεν έχει ανάγκη άλλους resolvers. Όταν δεν βρίσκει στην τοπική του cache την απάντηση σε κάποιο ερώτημα, ξεκινά την αναζήτηση από τους root name servers, μετά από τον κατάλληλο TLD server, ύστερα από τον domain-level server που θα του υποδειχθεί κ.ο.κ. Τις απαντήσεις που παίρνει δεν τις επιστρέφει απλά στους ενδιαφερόμενους πελάτες, αλλά ταυτόχρονα τις αποθηκεύει και στην cache του για μελλοντική αναφορά (και για όσο χρόνο επιτρέπουν τα αντίστοιχα TTL). Αν δεν θέλετε τα ερωτήματα των πελατών σας να φτάνουν σε δημόσιους caching name servers ή σε εκείνους του ISP σας, προτείνεται τότε να ρυθμίσετε το BIND ώστε να λειτουργεί ως recursive name server.

Forwarding. Όταν ένας name server της κατηγορίας δεν βρίσκει στην cache του την απάντηση σε ερώτημα πελάτη, τότε αμέσως προωθεί το ίδιο ερώτημα σε άλλους name servers που του έχουμε υποδείξει και είναι ικανοί για recursion. Με το που μαθαίνει ένας forwarding name server μιαν απάντηση, τη γνωστοποιεί στον ενδιαφερόμενο πελάτη αλλά την αποθηκεύει και στην cache του για μελλοντική χρήση. Είναι προφανές ότι ένας forwarding name server είναι σχεδιασμένος ώστε να αποφεύγει τη βαριά δουλειά. Υπό αυτή την έννοια, όταν το bandwidth που διαθέτουμε προς τον έξω κόσμο είναι περιορισμένο ή/και όταν η ποιότητα της σύνδεσης είναι χαμηλή, τότε η επιλογή ενός name server του είδους αποτελεί άριστη ιδέα. Θα εξαρτώμαστε βέβαια από άλλους recursive name servers, όπως, π.χ., είναι εκείνοι της Google, της OpenDNS ή του ISP μας (δεν τους προτείνουμε), ωστόσο ένας τέτοιος συμβιβασμός δεν είναι απαραίτητα τραγικός.

Τι ακολουθεί

Σε πολύ λίγο θα εγκαταστήσουμε το BIND σε ένα μηχάνημα με Ubuntu Server LTS 14.04 (http://www.ubuntu.com/server), το οποίο θα ρυθμίσουμε ώστε να λειτουργεί ως caching name server με ικανότητες για recursion. Βασιζόμενοι σ' αυτή τη διαμόρφωση θα δούμε πώς τροποποιείται, ώστε εναλλακτικά το BIND να λειτουργεί ως forwarding name server. Δεν θα παραλείψουμε να κάνουμε και μερικές δοκιμές, ώστε τέλος πάντων να δούμε πόσο πιο γρήγορος είναι –αν είναι– ο δικός μας name server, σε σύγκριση με έναν μεγάλο και σπουδαίο όπως, π.χ., είναι εκείνοι της Google ή της OpenDNS. Στο άρθρο εξάλλου που αρχίζει από τη σελίδα 46 του παρόντος, θα παρουσιάσουμε μερικές στρατηγικές εκμετάλλευσης του νέου name server. Θα εξετάσουμε, μ' άλλα λόγια, τρόπους ώστε κάποιες ή και όλες οι συσκευές του LAN να εξυπηρετούνται από τον καινούργιο name server.

Εγκατάσταση BIND

Από την κονσόλα του Ubuntu Server ή απομακρυσμένα, μέσω SSH, ενημερώνουμε πρώτα την τοπική λίστα με τα πακέτα από τα διαθέσιμα αποθετήρια:

```
~$ sudo apt-get update
```

Αμέσως μετά εγκαθιστούμε το BIND (bind9), ένα πακέτο με χρήσιμα εργαλεία (bind9utils), καθώς και τη συνοδευτική τεκμηρίωση (bind9-doc, προαιρετικό πακέτο):

```
~$ sudo apt-get install bind9 bind9utils bind9-doc
```

Αυτό ήταν. Όλα τα αρχεία ρυθμίσεων του BIND είναι συγκεντρωμένα κάτω από τον κατάλογο /etc/bind, οπότε ας μεταβούμε σ' αυτόν:

```
~$ cd /etc/bind
```

Το βασικό αρχείο ρυθμίσεων είναι το named.conf ("named" είναι το εναλλακτικό όνομα του BIND), το οποίο ως έχει δεν κάνει τίποτε άλλο από το να διαβάζει τα περιεχόμενα των αρχείων named.conf.default-zones, named.conf.local και named.conf.options. Στο παρόν άρθρο –και για τα είδη των name servers που μας ενδιαφέρουν– θα μας απασχολήσει μόνο το named.conf.options.

Recursive name server

Όπως προαναφέραμε το BIND είναι εξαιρετικά ευέλικτο κι αμέσως τώρα θα του πούμε να συμπεριφέρεται ως caching name server, με δυνατότητες recursion. Χωρίς άλλες περιστροφές ανοίγουμε το αρχείο named.conf.options με έναν text editor, όπως, π.χ., είναι το nano:

```
/etc/bind$ sudo nano named.conf.options
```

Ιδού πώς δείχνει το αρχείο *χωρίς* τα σχόλια:

```
options {
      directory "/var/cache/bind";
      dnssec-validation auto;
      auth-nxdomain no;
      listen-on-v6 { any; };
};
```

Πρώτη μας δουλειά είναι να ορίσουμε ένα ACL (Access Control List), δηλαδή μια λίστα, με τους πελάτες που επιτρέπεται να μιλούν με τον name server. Ο ορισμός αυτής της λίστας γίνεται στην αρχή του αρχείου named.conf.options κι έξω από το μπλοκ ονόματι "options". Τη λίστα την ονομάζουμε όπως θέλουμε:

```
acl goodfellas {
      10.0.0.0/24;
      localhost;
};
```

Επηρεασμένοι από την ταινία GoodFellas (http://www.imdb.com/title/tt0099685) που θυμηθήκαμε προσφάτως, ονομάσαμε τη λίστα μας goodfellas (όλα πεζά) και φροντίσαμε ώστε να περιλαμβάνει:

- όλα τα μηχανήματα με IP της μορφής 10.0.0.*, αφού αυτές είναι οι διευθύνσεις IP που παίρνουν οι clients του τοπικού μας δικτύου
- το τοπικό μηχάνημα (localhost), ώστε οι εφαρμογές του συστήματος να είναι σε θέση να χρησιμοποιούν κι αυτές τον name server

Σημειώστε ότι στη θέση των "10.0.0.0/24" και "127.0.0.1" κάλλιστα μπορούμε να βάλουμε το "localnets", το οποίο πάντα σημαίνει "το τοπικό μηχάνημα μαζί με τα δίκτυα κάθε network interface". Πιο συγκεκριμένα: Το μηχάνημά μας με το BIND διαθέτει μία μόνο κάρτα δικτύου Ethernet (ένα network interface), έχει διεύθυνση IP την 10.0.0.5 και μάσκα δικτύου την 255.255.255.0, επομένως για την περίπτωση του παραδείγματός μας το localnet σημαίνει "127.0.0.1 και 10.0.0.0/24" (Περισσότερα για τις διευθύνσεις IP, τις μάσκες δικτύου, τη μορφή CIDR κ.λπ., διαβάστε στο http://deltahacker.gr/hownetsworkpart2b). Το goodfellas, λοιπόν, μπορεί να οριστεί συντομότερα κι έτσι:

```
acl goodfellas {
      localnets;
};
```

Να πώς μοιάζει το named.conf.options έως αυτή τη στιγμή:

```
acl goodfellas {
      localnets;
};
      options {
      directory "/var/cache/bind";
      dnssec-validation auto;
      auth-nxdomain no;
      listen-on-v6 { any; };
};
```

Ο ορισμός του ACL δεν είναι αρκετός, ώστε τα queries από τα μηχανήματα που περιγράφει να γίνονται αποδεκτά. Απαιτείται επιπρόσθετα η οδηγία "allow-query { goodfellas; };", εντός του μπλοκ "options". Χρειάζεται εξάλλου να ζητήσουμε την ενεργοποίηση του recursion με την οδηγία "recursion yes;" – πάντα μέσα στο μπλοκ "options". Δείτε ξανά το named.conf.options, το οποίο πλέον περιλαμβάνει και τις δύο προαναφερθείσες οδηγίες:

```
acl goodfellas {
      localnets;
};
options {
      directory "/var/cache/bind";
      recursion yes;
      allow-query { goodfellas; };
      dnssec-validation auto;
      auth-nxdomain no;
      listen-on-v6 { any; };
};
```

Δεν χρειάζεται καμία άλλη ρύθμιση ώστε το BIND να λειτουργεί ως caching name server με ικανότητες recursion. Αν *δεν* θέλετε να συμπεριφέρεται ως forwarding name server, τότε προσπεράστε την επόμενη ενότητα και συνεχίστε με τη μεθεπόμενη. Διαφορετικά συνεχίστε το διάβασμα κανονικά.

Forwarding name server

Ξεκινάμε από το τροποποιημένο named.conf.options, ακριβώς όπως το αφήσαμε στην προηγούμενη ενότητα. Προσοχή: Επειδή φτιάχνουμε έναν forwarding name server, δεν σημαίνει ότι τώρα θ' αλλάξουμε την οδηγία "recursion yes;" σε "recursion no;". Αντιθέτως θα την αφήσουμε ως έχει, αφού ο name server συνεχίζει ν' απαντά για domains που δεν είναι στην ευθύνη του κι επομένως για τους πελάτες συνεχίζει να παρέχει recursive υπηρεσίες. Σίγουρα φορτώνει το μεγαλύτερο μέρος της δουλειάς σε άλλους name servers, αλλά αυτό δεν έχει σημασία και υπό μία έννοια *είναι* recursive. Και μιας κι αναφερθήκαμε σ' αυτούς τους άλλους servers, ας ορίσουμε μια λίστα με caching name servers στους οποίους θα προωθούνται τα αιτήματα των πελατών. Ο ορισμός λαμβάνει χώρα εντός του μπλοκ "options":

```
forwarders {
      8.8.8.8;
      8.8.4.4;
};
```

Η λίστα υλοποιήθηκε με το μπλοκ "forwarders" και περιλαμβάνει τους δύο δημόσιους name servers που απλόχερα παρέχει η Google. Θα μπορούσαμε να χρησιμοποιήσουμε άλλους, όπως, π.χ., εκείνους της OpenDNS (208.67.222.222 και 208.67.220.220) ή ακόμη και συνδυασμούς. Χρωστάμε και την οδηγία "forward only;", ώστε ο name server μας να προωθεί *πάντα* τα αιτήματα των πελατών και να μην επιχειρεί να τα εξυπηρετεί ο ίδιος (εκτός κι αν έχει τις απαντήσεις στην cache του, δηλαδή). Δείτε παρακάτω όλα τα περιεχόμενα του αρχείου named.conf.options για τον forwarding name server μας:

```
acl goodfellas {
      localnets;
};
options {
      directory "/var/cache/bind";
      recursion yes;
      allow-query { goodfellas; };
      forwarders {
         8.8.8.8;
         8.8.4.4;
      };
      forward only;
      dnssec-enable yes;
      dnssec-validation yes;
      auth-nxdomain no;
      listen-on-v6 { any; };
};
```

Οι παρατηρητικοί θα διαπιστώσατε ότι:

- το "dnssec-validation auto;" έχει γίνει "dnssec-validation yes;"
- από πάνω του έχει προστεθεί η οδηγία "dnssec-enable yes;»

Χωρίς αυτές τις δύο αλλαγές, οι ρυθμίσεις DNSSEC των name servers στους οποίους προωθούμε τα αιτήματα θα μολύνουν τα log files του server μας με μηνύματα λάθους.

Έλεγχος ορθής λειτουργίας κι ενεργοποίηση

Αποθηκεύουμε τις αλλαγές στο αρχείο named.conf.options (στο nano, π.χ., δίνουμε [CTRL+O], [Enter]) κι αμέσως εγκαταλείπουμε τον editor (π.χ., με [CTRL+X] στο nano). Έχοντας δικαιώματα root ανοίγουμε στα γρήγορα το αρχείο ρυθμίσεων του BIND daemon, το /etc/default/bind9:

```
/etc/bind$ sudo nano /etc/default/bind9
```

Φροντίζουμε ώστε η γραμμή

```
OPTIONS="-u bind"
```

να γίνει

```
OPTIONS="-u bind -4"
```

Αποθηκεύουμε την αλλαγή κι εγκαταλείπουμε τον editor. Μόλις προσθέσαμε αυτό το "-4» και, ως αποτέλεσμα, το BIND σε λίγο θα δέχεται μόνο αιτήματα για resolving διευθύνσεων IPv4. Στο Ubuntu, εξάλλου, η υπηρεσία ενεργοποιείται αμέσως μετά την εγκατάσταση του αντίστοιχου πακέτου (bind9). Αυτή τη στιγμή βέβαια δεν έχουν ληφθεί υπόψη οι τροποποιήσεις στο /etc/bind/named.conf.options και, πριν επανεκκινήσουμε το BIND ώστε να τις διαβάσει, είναι καλή ιδέα να ελέγξουμε το named.conf.options για συντακτικά λάθη. Προς τούτο θα καταφύγουμε στο εργαλείο named-checkconf, το οποίο βρισκόταν στο πακέτο bind9utils που εγκαταστήσαμε νωρίτερα:

```
/etc/bind$ sudo named-checkconf
```

Αν δεν υπάρχει λάθος, τότε το παραπάνω δεν θα επιστρέψει κάτι. Ας επανεκκινήσουμε λοιπόν το BIND:

```
/etc/bind$ sudo service bind9 restart
* Stopping domain name service... bind9
waiting for pid 1341 to die
                                          [ OK ]
* Starting domain name service... bind9   [ OK ]
/etc/bind$
```

Πολύ ωραία. Ένας τρόπος για να βεβαιωθούμε ότι το BIND αφουγκράζεται για αιτήματα πελατών, είναι με τη βοήθεια του netstat:

```
/etc/bind$ sudo netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 10.0.0.5:53             0.0.0.0:*               LISTEN      1622/named
```

```
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      1622/named
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1312/sshd
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN      1622/named
tcp        0     36 10.0.0.5:22             10.0.0.104:60922        ESTABLISHED 1483/sshd: admin [p
tcp6       0      0 :::22                   :::*                    LISTEN      1312/sshd
/etc/bind$
```

Παρατηρούμε ότι το BIND δέχεται αιτήματα από το port 53/TCP και, όπως βλέπετε, στο σύστημά μας τα αναμένει από το network interface με IP το 10.0.0.5, αλλά κι από το loopback interface με IP το 127.0.0.1.

Δοκιμές και σύγκριση επιδόσεων

Τις πρώτες μας δοκιμές μπορούμε να τις κάνουμε από μια κονσόλα του ίδιου του Ubuntu Server με τη βοήθεια του εργαλείου dig, το οποίο παρέχεται από το πακέτο dnsutils που είναι ήδη εγκατεστημένο. Ας ζητήσουμε από τον τοπικό name server το IP του colder.xyz:

```
~$ dig @127.0.0.1 colder.xyz

; <<>> DiG 9.9.5-3ubuntu0.5-Ubuntu <<>> colder.xyz
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12027
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;colder.xyz.       IN A
;; ANSWER SECTION:
colder.xyz.    1799   IN A   46.101.173.140

;; Query time: 290 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Sep 27 09:34:07 EEST 2015
;; MSG SIZE  rcvd: 55
```

Προκειμένου να μην ερωτηθεί ο name server που ήδη γνωρίζει το μηχάνημα αλλά ο τοπικός, στο dig δώσαμε την οδηγία "@127.0.0.1». Το BIND δεν είχε το IP του colder.xyz στην cache του, το αναζήτησε, το βρήκε (είναι το 46.101.173.140), το αποθήκευσε

στην cache του και το επέστρεψε στον πελάτη (που σ' αυτή την περίπτωση είναι το dig). Η διαδικασία της αναζήτησης διήρκησε 290 χιλιοστά του δευτερολέπτου (290 msec, βλ. γραμμή με το Query time). Ας ρωτήσουμε ξανά το BIND, πάλι για το colder.xyz:

```
~$ dig @127.0.0.1 colder.xyz

; <<>> DiG 9.9.5-3ubuntu0.5-Ubuntu <<>> @127.0.0.1 colder.xyz
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21003
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;colder.xyz.      IN A
;; ANSWER SECTION:
colder.xyz.    1246  IN A  46.101.173.140

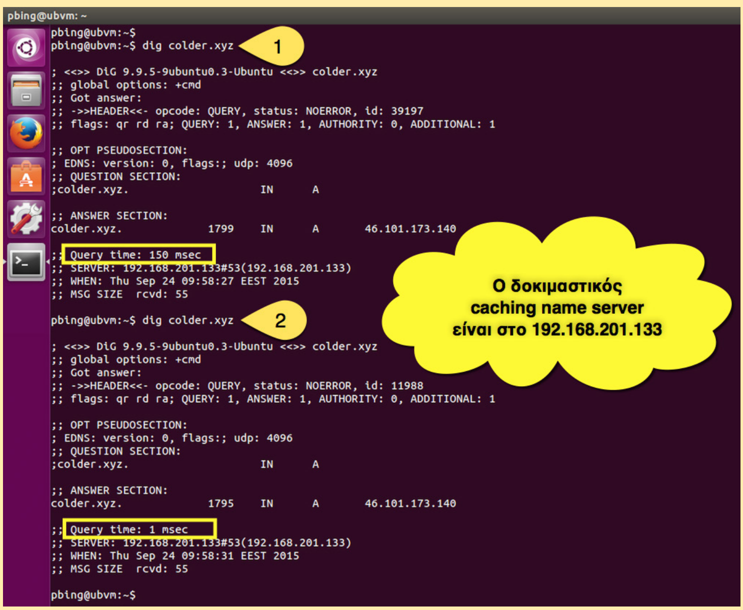
;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Sep 27 09:43:20 EEST 2015
;; MSG SIZE  rcvd: 55
```

Αυτή τη φορά το BIND είχε τη ζητούμενη πληροφορία στην cache του, οπότε το ερώτημα εξυπηρετήθηκε σε μηδενικό, πρακτικά, χρόνο! (Query time: 0 msec) Τώρα, για να εκτιμήσουμε πόσο μεγάλη βελτίωση στις επιδόσεις προσφέρει ένας τοπικός name server, ας απευθύνουμε το ίδιο ερώτημα διαδοχικά σε έναν εξωτερικό, όπως, π.χ., είναι ο 8.8.8.8 της Google. Θα καταφύγουμε και πάλι στο dig αλλά για οικονομία χώρου θα φιλτράρουμε την έξοδό του με το grep, ώστε να βλέπουμε μόνο τη γραμμή με το "Query time":

```
~$ dig @8.8.8.8 colder.xyz | grep "Query time"
;; Query time: 301 msec
~$ dig @8.8.8.8 colder.xyz | grep "Query time"
;; Query time: 85 msec
~$ dig @8.8.8.8 colder.xyz | grep "Query time"
;; Query time: 124 msec
~$ dig @8.8.8.8 colder.xyz | grep "Query time"
```

```
;; Query time: 116 msec
~$ dig @8.8.8.8 colder.xyz | grep "Query time"
;; Query time: 73 msec
~$
```

Αμέσως μετά το πρώτο ερώτημα, το οποίο απαντήθηκε σε 301 msec, ο name server της Google αποθηκεύει στην cache του το IP του colder.xyz και στα επόμενα ερωτήματα απαντά γρηγορότερα (85 msec, 124 msec, 116 msec, 73 msec κ.ο.κ.) Παρά το γεγονός ότι πρόκειται για πανίσχυρο μηχάνημα, λόγω της απόστασής του από το δίκτυό μας είναι αδύνατον να συναγωνιστεί τον τοπικό μας name server!



Τον νέο μας name server καλό είναι να μην τον δοκιμάζουμε μόνο τοπικά, αλλά κι από κάποιο άλλο μηχάνημα του τοπικού δικτύου. Στο screenshot φαίνεται μέρος των ελέγχων που κάναμε στο πλαίσιο της προετοιμασίας του παρόντος άρθρου. Ο name server βρισκόταν στη διεύθυνση 192.168.201.133, ενώ ο υπολογιστής από τον οποίο εργαζόμασταν είχε ήδη ενημερωθεί για την παρουσία του. Παρατηρήστε τη διαφορά στους χρόνους απόκρισης μεταξύ δύο διαδοχικών ερωτημάτων για το ίδιο domain: Την πρώτη φορά η απάντηση ήλθε σε 150 msec, ενώ τη δεύτερη 150 φορές γρηγορότερα.

Η συνέχεια

Στο σημείο αυτό ο name server μας πρέπει να λειτουργεί σωστά και να 'ναι πανέτοιμος για δράση. Γυρίστε τώρα στο άρθρο που αρχίζει από τη σελίδα 46 του τεύχους, όπου παρουσιάζουμε διάφορους τρόπους προκειμένου τα μηχανήματα του τοπικού μας δικτύου να εξυπηρετούνται από το BIND του Ubuntu Server.

Skill: Intermediate
Tags: .NET, C#, programming, desktop, registry

Ανάπτυξη εφαρμογών σε C# για το .NET [μέρος 9ο]

Μέσα από ένα μακρύ αλλά διασκεδαστικό ταξίδι στον κόσμο της C#, καταφέραμε να δημιουργήσουμε μια πλήρη και λειτουργική εφαρμογή για το περιβάλλον των Windows: τον text editor ονόματι deltaPad! Το μικρόβιο του deltaHacker, όμως, δεν μάς αφήνει να επαναπαυθούμε. Συνεχίζουμε λοιπόν με στόχο την κατάκτηση της επόμενης κορυφής του .NET framework.

του Πέτρου Κυλαδίτη

Δεν ξέρουμε για εσάς αλλά εμείς, στο Eastern Outpost της Parabing, συλλάβαμε πολλές φορές τους εαυτούς μας να χρησιμοποιούμε το deltaPad αντί του κλασικού σημειωματάριου των Windows (βλ. το σχετικό άρθρο στο deltaHacker 046). Δεν μπορούσαμε να προσπεράσουμε τη σκέψη ότι πρόκειται για καρπό των δικών μας προσπαθειών, ενώ ήταν στη μέση και το πείσμα. Θέλαμε να αποδείξουμε την ανωτερότητα της δικής μας εφαρμογής έναντι του Notepad, καθώς και το γεγονός ότι η ενασχόληση με τη C# άξιζε τον κόπο. Κάπως έτσι, δεν πέρασε πολύς χρόνος έως ότου θέσουμε τον επόμενό μας στόχο: την περαιτέρω βελτίωση του deltaPad! Αυτή η προσπάθεια εκτός από διασκεδαστική, θα ήταν και ενδιαφέρουσα, αφού θα μας έφερνε σε επαφή με ακόμα πιο προχωρημένες τεχνικές του .NET framework.

Σώζοντας τον κόπο μας

Κάτι που μας ενοχλούσε ιδιαίτερα στο παλιό deltaPad ήταν η απουσία μιας ερώτησης κατά τον τερματισμό της εφαρμογής, για την αποθήκευση των όποιων τροποποιήσεων στο ανοιγμένο αρχείο. Για να λύσουμε αυτό το πρόβλημα σκεφτήκαμε να δημιουργήσουμε μια μεταβλητή που θα λειτουργεί ως σημαία (flag) και θα επισημαίνει το αν υπάρχουν αλλαγές στο τρέχον κείμενο. Η εν λόγω σημαία θα σηκώνεται με οποιαδήποτε αλλαγή στο κείμενο και θα κατεβαίνει αυτόματα με την αποθήκευση. Η σχετική μεταβλητή θα ελέγχεται από το πρόγραμμα κατά το κλείσιμο του deltaPad ή κατά τη φόρτωση ενός νέου αρχείου, ενέργεια που ως συνέπεια έχει το κλείσιμο του προηγούμενου. Όταν διαπιστώνεται ότι υπάρχουν αλλαγές, το πρόγραμμα θα εμφανίζει στο χρήστη μια ερώτηση για την αποθήκευση του αρχείου ή την ακύρωση της ενέργειας και την παραμονή στο ίδιο έγγραφο.

Ξεκινάμε, λοιπόν, ορίζοντας εντός της κλάσης MainForm μια κατάλληλη μεταβλητή:

```
private bool textChanged = false;
```

Προφανώς, όταν ξεκινά το πρόγραμμα η τιμή της textChanged δεν μπορεί παρά να είναι false, εφόσον είναι αδύνατο να έχουμε τροποποιήσει ήδη κάποιο αρχείο. Έτσι, την αρχικοποιούμε σε false κατά την εκκίνηση του προγράμματος (φόρτωση MainForm) και την αλλάζουμε σε true οποτεδήποτε τροποποιείται το κείμενο. Η οποιαδήποτε αλλαγή στο κείμενο ενός textbox πυροδοτεί το συμβάν TextBoxTextChanged. Εμείς εκμεταλλευόμαστε αυτό το συμβάν για την ενημέρωση της textChange ως εξής:

```
void TextBoxTextChanged(object sender, EventArgs e) {
    if(!textChanged)
        textChanged = true;
}
```

Εφόσον δεν χρειάζεται να θέτουμε την τιμή της textChanged σε true διαρκώς, δηλαδή κάθε φορά που τροποποιείται το κείμενο, αρκεί να ελέγχουμε την τιμή της κι αν είναι ήδη true να παρακάμπτουμε την εκχώρηση. Με αυτές τις λίγες γραμμές ολοκληρώνεται ο μηχανισμός της σημαίας που μας ειδοποιεί για τυχόν αλλαγές στο κείμενο. Σειρά έχουν οι απαραίτητοι έλεγχοι της μεταβλητής textChanged, καθώς και η εμφάνιση της σχετικής ερώτησης. Μια προφανής θέση γι' αυτόν τον έλεγχο είναι το κλείσιμο του προγράμματος και άλλη μία αποτελεί το άνοιγμα ενός αρχείου. Ακολουθώντας τις καλές πρακτικές προγραμματισμού, δημιουργούμε μια μέθοδο που θα εξυπηρετεί τις ανάγκες μας και στις δύο περιπτώσεις. Δείτε τη μέθοδο AskSave:

```
private DialogResult AskSave() {
    DialogResult result = MessageBox.Show(
         "Θέλετε να αποθηκεύσετε τις αλλαγές που πραγματοποιήθηκαν;",
         "Το κείμενο έχει τροποποιηθεί",
         MessageBoxButtons.YesNoCancel,
         MessageBoxIcon.Question);
    if(result == DialogResult.Yes) {
        SaveText();
    }
    return result;
}
```

Η παραπάνω μέθοδος δεν δέχεται παραμέτρους και εμφανίζει ένα πλαίσιο μηνύματος με τρεις δυνατές απαντήσεις: "Ναι", "Όχι" και "Άκυρο". Η μέθοδος MessageBox.Show() είναι υπεύθυνη για τη σχεδίαση και την εμφάνιση του πλαισίου με την ερώτηση. Επιστρέφει μια απαρίθμηση τύπου DialogResult, από την οποία μπορούμε να

παίρνουμε την απάντηση του χρήστη. Έτσι, στην περίπτωση που ο χρήστης πατήσει το κουμπί "Ναι", καλούμε τη μέθοδο SaveText(). Πρόκειται για μια μέθοδο που είχαμε δημιουργήσει στο προηγούμενο μέρος της σειράς και η οποία αναλαμβάνει την αποθήκευση του κειμένου. Στο τέλος, η AskSave επιστρέφει την επιλογή του χρήστη (την απαρίθμηση που πήραμε από τη MessageBox.Show), καθώς η επιλογή του "Άκυρο" απαιτεί διαφορετικό χειρισμό (ανάλογα με την περίσταση).

Ας εξετάσουμε τώρα το σημείο τερματισμού της εφαρμογής. Ο τερματισμός συνεπάγεται το κλείσιμο της κύριας φόρμας, γεγονός που πυροδοτεί τα συμβάντα FormClosing (η φόρμα κλείνει αλλά είναι ακόμα ορατή) και FormClosed (η φόρμα έκλεισε και δεν υπάρχει στην οθόνη). Εμείς θα χρησιμοποιήσουμε το πρώτο συμβάν, αφού δεν θέλουμε να διαγραφεί η φόρμα πριν αποφασίσει ο χρήστης τι προτίθεται να κάνει.

```
void MainFormFormClosing(object sender, FormClosingEventArgs e) {
    if(textChanged) {
        if(AskSave() == DialogResult.Cancel) {
            e.Cancel = true;
        }
    }
}
```

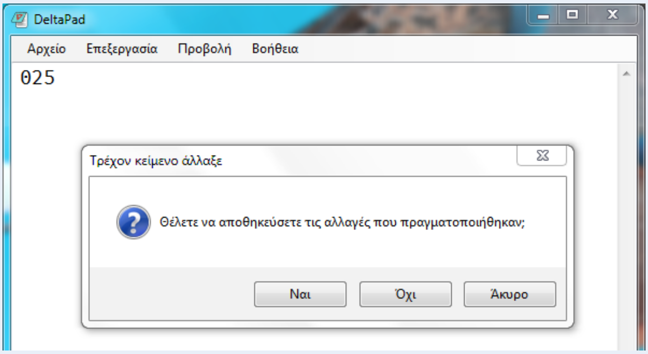
Όταν η μεταβλητή textChanged έχει την τιμή true (το κείμενο έχει αλλάξει), καλούμε τη συνάρτηση AskSave() κι αναμένουμε το αποτέλεσμά της. Αν αυτό τυγχάνει να είναι DialogResult.Cancel, φροντίζουμε να ακυρώσουμε το κλείσιμο της φόρμας τροποποιώντας κατάλληλα τις παραμέτρους του συμβάντος του κλεισίματος (FormClosingEventArgs). Συγκεκριμένα, θέτουμε την ιδιότητα cancel σε true, που θα έχει σαν συνέπεια να αποτραπεί ο τερματισμός του προγράμματος.

Το άλλο σημείο όπου θα θέλουμε να αξιοποιήσουμε τη μέθοδο AskSave είναι το άνοιγμα ενός αρχείου. Για το λόγο αυτό επεμβαίνουμε στην υπάρχουσα μέθοδο LoadText, ως εξής:

```
private void LoadText() {
    if(textChanged) {
        if(AskSave() == DialogResult.Cancel) {
            return;
        }
    }
    DialogResult result = openFileDialog.ShowDialog(this);
    if(result == DialogResult.OK) {
        try{
            textBox.Text = File.ReadAllText(openFileDialog.FileName);
```

```
                textChanged = false;
            }catch(Exception err) {
                MessageBox.Show(err.Message,
                    "Σφάλμα φόρτωσης αρχείου",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
    }
```

Εδώ, αν η AskSave επιστρέψει την τιμή DialogResult.Cancel τότε η φόρτωση του αρχείου τερματίζεται με την εκτέλεση της return. Επίσης, προσέξτε ότι μετά τη φόρτωση του κειμένου στο textbox ορίζουμε την textChanged σε false, αφού πλέον έχουμε μπροστά μας ένα νέο κείμενο.



Κλείνοντας την εφαρμογή εμφανίζεται πλέον μια σχετική προειδοποίηση, επιτρέποντας στο χρήστη να αποθηκεύσει όσες αλλαγές ενδέχεται να πραγματοποίησε στο ανοιγμένο αρχείο.

Αποθήκευση επιλογών

Κάθε πρόγραμμα που σέβεται τον εαυτό του διατηρεί όλες τις αλλαγές που κάνει ο χρήστης στις προεπιλεγμένες ρυθμίσεις. Γιατί όχι και το deltaPad; Άλλωστε ακόμα κι εμείς οι ίδιοι, οι περήφανοι δημιουργοί του, εκνευριζόμασταν με την ανάγκη να αλλάζουμε συχνά πυκνά τη γραμματοσειρά, το μέγεθος του παραθύρου κ.ο.κ. Θα θέλαμε να κάνουμε τις σχετικές ρυθμίσεις και το πρόγραμμα να τις θυμάται μέχρι να αλλάξουμε γνώμη.

Οι λύσεις για την αποθήκευση των ρυθμίσεων είναι πολλές. Η πιο απλή περιλαμβάνει τη χρήση αρχείων απλού κειμένου. Αυτά τα αρχεία περιλαμβάνουν μια ρύθμιση σε κάθε γραμμή και το περιεχόμενό τους φορτώνεται σε έναν πίνακα κατά την εκκίνηση του προγράμματος. Μια άλλη λύση σχετίζεται με τα αρχεία INI, ενώ μια τρίτη προβλέπει τη χρήση αρχείων XML. Παρεμπιπτόντως, η τρίτη λύση προτείνεται κι

από την Microsoft για την πλατφόρμα .NET. Η συγκεκριμένη λύση φαίνεται δελεαστική αλλά μας προκαλεί τρομερή ναυτία, καθώς οι κλάσεις που προσφέρει το .NET γι' αυτή τη δουλειά είναι επιεικώς για τα μπάζα. Πάντως, εκτός από την αποθήκευση των ρυθμίσεων σε κάποιο αρχείο υπάρχει πάντα και η λύση του μητρώου των Windows. Το γνωστό μας Registry, όπως αρεσκόμαστε να το λέμε εδώ στην περιοχή της Δασκαλόπετρας στη Χίο (δίπλα ακριβώς από το ιερό της Κυβέλης, όπου ο Όμηρος εξιστορούσε τα ομώνυμα έπη στους μαθητές του), έχει μια πολύ μεγάλη ιστορία που ξεκινά από την έκδοση 3.1 των Windows, ενώ συνοδεύεται κι από μια μυστηριακή φήμη. Στην πράξη, το μητρώο δεν είναι τίποτα άλλο παρά μια (ιεραρχικά δομημένη) βάση δεδομένων. Μπορούμε να το φανταζόμαστε σαν ένα σύνολο φακέλων, μέσα στους οποίους υπάρχουν άλλοι φάκελοι και αρχεία, τα οποία φυλάσσουν πληροφορίες/ρυθμίσεις. Στην ορολογία του μητρώου, οι υποφάκελοι ονομάζονται κλειδιά και τα αρχεία τιμές. Επομένως, ένα κλειδί μπορεί να περιλαμβάνει πολλά υποκλειδιά και πολλές τιμές. Μέσα στο μητρώο υπάρχουν κάποια βασικά κλειδιά, κάτω από τα οποία αποθηκεύονται γενικές ρυθμίσεις του συστήματος. Εμείς, για την αποθήκευση των ρυθμίσεων του προγράμματός μας μπορούμε να στραφούμε στο κλειδί HKEY_CURRENT_USER. Εκεί αποθηκεύονται ρυθμίσεις που αφορούν αποκλειστικά στον τρέχοντα χρήστη. Μεταξύ άλλων υπάρχει και το υποκλειδί SOFTWARE, που περιλαμβάνει ρυθμίσεις για τα προγράμματα του συγκεκριμένου χρήστη. Κάθε φορά που σκοπεύουμε να αποθηκεύσουμε πολλές ρυθμίσεις που σχετίζονται μεταξύ τους, μια καλή πρακτική είναι να τις τοποθετούμε "κάτω" από το ίδιο κλειδί. Έτσι και για τις ανάγκες του deltaPad, μπορούμε να δημιουργήσουμε τα εξής κλειδιά:

```
HKEY_CURRENT_USER\Software\deltaHacker\deltaPad
```

Για τη διαχείριση του μητρώου το .NET framework παρέχει την κλάση RegistryKey, η οποία ανήκει στο χώρο ονομάτων Microsoft.Win32. Επομένως, αρκεί να εισάγουμε το σχετικό χώρο ονομάτων στον κώδικά μας και να δημιουργήσουμε μια μέθοδο η οποία θα κατασκευάζει όλα τα απαραίτητα κλειδιά που αναφέραμε νωρίτερα:

```
private RegistryKey ApplicationRegKey() {
    RegistryKey key = Registry.CurrentUser.OpenSubKey("Software",true);
    key.CreateSubKey(“deltaHacker”);
    key = key.OpenSubKey(Application.CompanyName, true);
    key.CreateSubKey(“deltaPad”);
    key = key.OpenSubKey(Application.ProductName, true);
    return key;
}
```

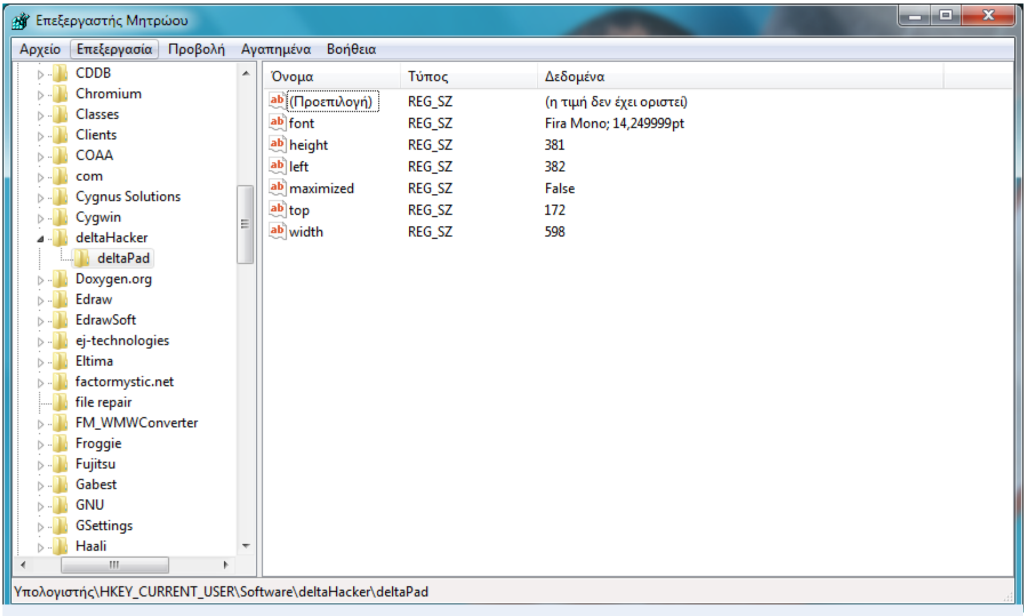
Η παραπάνω μέθοδος επιστρέφει ένα "ανοιγμένο" κλειδί, έτοιμο να δεχτεί την όποια επεξεργασία. Για να αποθηκεύσουμε μια τιμή μπορούμε να χρησιμοποιήσουμε το ανοιγμένο κλειδί ως εξής:

```
RegistryKey key = ApplicationRegKey();
key.SetValue("width", this.Width);
```

Στη συνέχεια, για να διαβάσουμε την αποθηκευμένη τιμή μπορούμε να κάνουμε κάτι τέτοιο:

```
int formWidth = (int) key.GetValue("width").ToString();
key.Close();
```

Αυτό που θα πρέπει να προσέχουμε κατά την εργασία με τα κλειδιά, είναι ότι στο τέλος πρέπει να τα "κλείνουμε"!



Οι ρυθμίσεις της εφαρμογής που έχουν αποθηκευτεί στο μητρώο, όπως φαίνονται μέσα από το εργαλείο regedit των Windows.

Αν η εργασία με το μητρώο σάς φαίνεται κουραστική, πρέπει να σας πούμε ότι δεν είσαστε μόνοι. Δεν χρειάζεται όμως να αποθαρρυνόμαστε και να αποφεύγουμε την αξιοποίηση του μητρώου, καθώς οι διάφοροι ανεξάρτητοι προγραμματιστές, ανάμεσά τους και ο γράφων, έχουν αναπτύξει κλάσεις που απλοποιούν στο μέγιστο αυτή την εργασία.

Για τις ανάγκες του άρθρου μας, λοιπόν, καταφεύγουμε στη χρήση της βιβλιοθήκης Multipetros.Confing.dll, η οποία διανέμεται υπό την άδεια FreeBSD και μπορείτε να την κατεβάσετε, μαζί με τον πηγαίο κώδικα και τη σχετική τεκμηρίωση, από τη διεύθυνση http://www.multipetros.gr/public-projects/libraries/confing-dll. Η συγκεκριμένη βιβλιοθήκη, εκτός από την κλάση RegIni, η οποία απλοποιεί την αποθήκευση

ρυθμίσεων στο μητρώο, παρέχει και τις κλάσεις Ini και SimpleIni, οι οποίες πετυχαίνουν κάτι παρόμοιο χρησιμοποιώντας αρχεία INI. Αφού εισάγουμε την βιβλιοθήκη στο έργο μας, εισάγουμε και τον αντίστοιχο χώρο ονομάτων στην αρχή του κώδικα της φόρμας:

```
using Multipetros.Config;
```

Στη συνέχεια δημιουργούμε μια μεταβλητή που την ονομάζουμε properties, μέσω της οποίας θα ανακτούμε και θα γράφουμε τις ρυθμίσεις στο μητρώο:

```
private RegistryIni properties = new RegistryIni(
                        Application.CompanyName,
                        Application.ProductName);
```

Παρατηρήστε ότι χρησιμοποιούμε τις ιδιότητες CompanyName και ProductName του αντικειμένου Application. Το συγκεκριμένο αντικείμενο αντιπροσωπεύει την εφαρμογή μας και οι δύο ιδιότητες, όπως φανερώνει και τ' όνομά τους, επιστρέφουν το όνομα της εταιρείας ανάπτυξης και το όνομα του προϊόντος. Αυτά τα δύο στοιχεία μπορούμε να τα ορίσουμε στο αρχείο AssemblyInfo.cs, που βρίσκεται κάτω από το στοιχείο Properties του παράθυρου Projects. Εξετάζοντας τις γραμμές του εν λόγω αρχείου θα βρούμε τις καταχωρήσεις AssemblyCompany και AssemblyProduct. Από εκεί πρέπει να δώσουμε στις συγκεκριμένες ιδιότητες μια ταιριαστή τιμή:

```
[assembly: AssemblyCompany("deltaHacker")]
[assembly: AssemblyProduct("deltaPad")]
```

Τώρα, θα εξετάσουμε μια μέθοδο που θα φορτώνει τις ιδιότητες από το μητρώο και θα τις εφαρμόζει στο πρόγραμμα. Η συγκεκριμένη μέθοδος θα χρησιμοποιείται κατά την εκκίνηση της εφαρμογής:

```
private void LoadProperties() {
    string valStr;
    int valInt;
    valStr = properties["width"];
    int.TryParse(valStr, out valInt);
    if(valInt == 0)
        valInt = 450;
    this.Width = valInt;

    valStr = properties["height"];
```

```
    int.TryParse(valStr, out valInt);
    if(valInt == 0)
        valInt = 300;
    this.Height = valInt;
    valStr = properties["top"];
    int.TryParse(valStr, valInt);
    if(valInt != 0)
        this.Top = valInt;
    valStr = properties["left"];
    int.TryParse(valStr, valInt);
    if(valInt != 0)
        this.Left = valInt;
    valStr = properties["maximized"];
    this.WindowState = (valStr == "True") ? FormWindowState.Maximized :
FormWindowState.Normal;
    valStr = properties["font"];
    if(valStr != "") {
        FontConverter fontCon = new FontConverter();
        Font storedFont = (Font) fontCon.ConvertFromString(valStr);
        textBox.Font = storedFont;
    }
}
```

Ας δούμε και τον κώδικα για μια μέθοδο που θα αποθηκεύει τις ιδιότητες στο μητρώο. Η κλήση αυτής της μεθόδου θα πραγματοποιείται κατά τον τερματισμό του προγράμματος.

```
private void SaveProperties() {
    if(this.WindowState == FormWindowState.Maximized) {
        properties["maximized"] = true.ToString();
    } else {
        properties["width"] = this.Width.ToString();
        properties["height"] = this.Height.ToString() ;
        properties["top"] = this.Top.ToString();
        properties["left"] = this.Left.ToString();
    }
    FontConverter fontCon = new FontConverter();
    properties["font"] = fontCon.ConvertToString(textBox.Font);
}
```

Στη μέθοδο LoadProperties δεν κάνουμε τίποτε άλλο από το να διαβάζουμε διαδοχικά τις τιμές που έχουν αποθηκευτεί στο μητρώο από το πρόγραμμά μας. Οι εν λόγω τιμές αποθηκεύονται ως συμβολοσειρές στη μεταβλητή valStr, ενώ ο αντίστοιχος ακέραιος –σε όσες περιπτώσεις πραγματοποιείται αυτή η μετατροπή– αποθηκεύεται στη μεταβλητή valInt. Αν κάποια από τις τιμές που προσπαθούμε να διαβάσουμε δεν υπάρχει στο μητρώο, λαμβάνουμε ένα κενό string. Η μετατροπή των συμβολοσειρών που αναπαριστούν αριθμούς πραγματοποιείται με τη μέθοδο TryParse, του αντικειμένου int. Η ίδια η μέθοδος επιστρέφει true όταν η μετατροπή πραγματοποιείται χωρίς πρόβλημα, ενώ σε κάθε άλλη περίπτωση επιστρέφει false. Το αποτέλεσμα της μετατροπής επιστρέφεται σε μια μεταβλητή που προσδιορίζουμε εμείς κατά την κλήση της TryParse. Όπως βλέπετε, οι πρώτες τιμές που επιχειρεί να διαβάσει από το μητρώο η μέθοδός μας αφορούν στις διαστάσεις και στη θέση της κύριας φόρμας του προγράμματος. Για να λειτουργήσει όπως αναμένουμε ο καθορισμός αυτών των τιμών, πρέπει να έχουμε θέσει την ιδιότητα StartPosition της φόρμας σε Manual. Στη συνέχεια, για τη τιμή maximized περιμένουμε ένα λεκτικό True ή False και, όπως βλέπετε, η συγκεκριμένη τιμή καθορίζει την κατάσταση της φόρμας. Τέλος, για τη φόρτωση της αποθηκευμένης γραμματοσειράς χρησιμοποιούμε το αντικείμενο FontConverter, το οποίο παρέχει τη μέθοδο ConvertFromString. Η συγκεκριμένη μέθοδος επιστρέφει ένα αντικείμενο Font, το οποίο έχει εκφραστεί ως συμβολοσειρά από την αντίστοιχη μέθοδο ConvertToString. Αυτή η μετατροπή, βέβαια, πραγματοποιείται στη μέθοδο που αποθηκεύει τις ρυθμίσεις μας. Τέλος, το αντικείμενο Font που λαμβάνουμε το αποδίδουμε ως τιμή στην αντίστοιχη ιδιότητα του textbox.

Στη μέθοδο SaveProperties πραγματοποιείται η αντίστροφη μετακίνηση δεδομένων: από το πρόγραμμα προς το μητρώο. Αρχικά ελέγχουμε την ιδιότητα WindowState. Αν η συγκεκριμένη ιδιότητα έχει την τιμή Maximized, αποθηκεύουμε στο μητρώο τη σχετική ρύθμιση και προσπερνάμε την αποθήκευση των υπόλοιπων τιμών. Όταν θα ξανατρέξουμε το πρόγραμμα, βλέπετε, θα αρκεί να μεγιστοποιήσουμε το παράθυρό του. Αν η ιδιότητα WindowState δεν έχει την τιμή Maximized, αποθηκεύουμε αυτή τη ρύθμιση, όπως επίσης τις διαστάσεις και τη θέση του παραθύρου. Φυσικά, οι σχετικές τιμές μετατρέπονται πρώτα σε συμβολοσειρές. Τέλος, όπως αναφέραμε και προηγουμένως, για την αποθήκευση της επιλεγμένης γραμματοσειράς χρησιμοποιούμε τη μέθοδο ConvertToString.

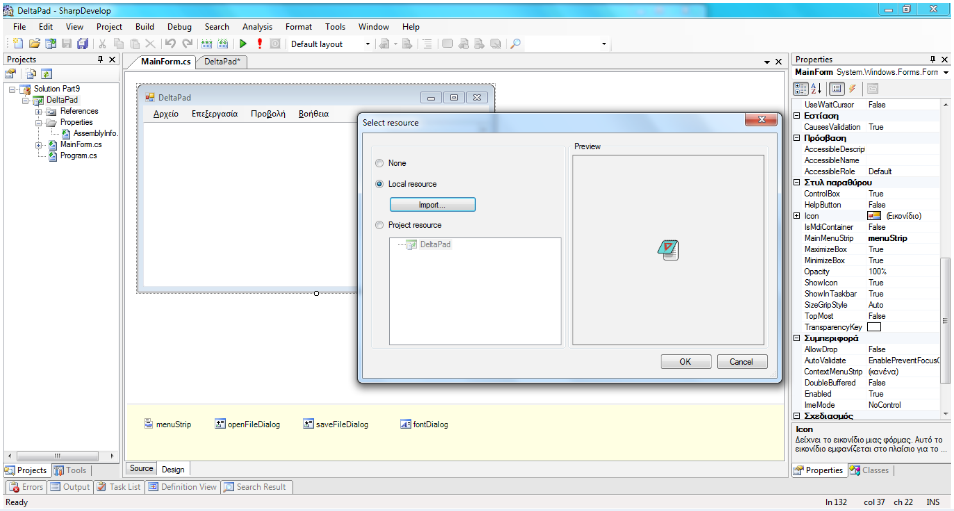
Το μόνο που απομένει τώρα είναι να φτιάξουμε δύο μεθόδους για το χειρισμό των γεγονότων Load και Closed της φόρμας μας. Το πρώτο πυροδοτείται κατά τη φόρτωση της φόρμας (εκκίνηση του προγράμματος) και το δεύτερο όταν πια η φόρμα έχει κλείσει (τερματισμός του προγράμματος):

```
void MainFormLoad(object sender, EventArgs e) {
    LoadProperties();
}
void MainFormFormClosed(object sender, FormClosedEventArgs e) {
    SaveProperties();
}
```

Μετά κι από αυτό μπορούμε να απολαύσουμε την ευκολία που προσφέρει η αυτόματη αποθήκευση των ρυθμίσεών μας. Μπορούμε να τρέξουμε το πρόγραμμα, να αλλάξουμε γραμματοσειρά και να μεταβάλλουμε τη θέση ή και το μέγεθος του παραθύρου. Την επόμενη φορά που θα το τρέξουμε θα το βρούμε στην ίδια κατάσταση που το αφήσαμε.

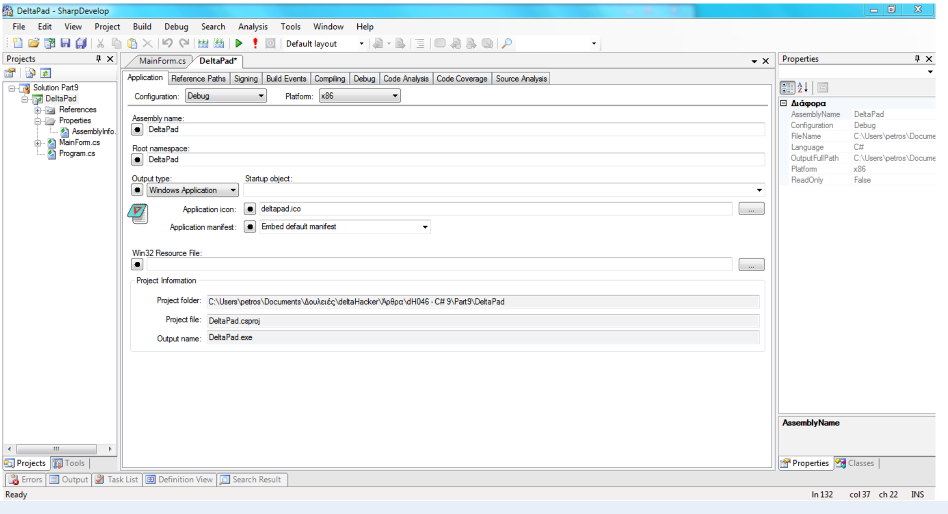
Το κερασάκι στην τούρτα

Μετά από τόσο κόπο πρέπει να φροντίσουμε ώστε το deltaPad να κάνει αισθητή την παρουσία του στην επιφάνεια εργασίας. Κάτι τέτοιο μπορεί να γίνει με ένα καλοσχεδιασμένο εικονίδιο! Εμείς φτιάξαμε ένα χρησιμοποιώντας τον αγαπημένο μας επεξεργαστή εικόνας (GIMP) και ένα εργαλείο επεξεργασίας εικονιδίων (Pixelformer). Όταν ετοιμάσουμε το εικονίδιό μας, καθώς και αρκετά στιγμιότυπα διαφορετικής ανάλυσης, μεταβαίνουμε στο παράθυρο Projects του SharpDevelop και κάνουμε διπλό κλικ πάνω στο Project DeltaPad. Από το μενού που εμφανίζεται επιλέγουμε το Properties κι έτσι εμφανίζεται ένα νέο παράθυρο με πολλές καρτέλες. Στην πρώτη από αυτές (Application) συναντάμε την καταχώριση Application Icon, από την οποία μπορούμε να δηλώσουμε το επιθυμητό εικονίδιο. Παρεμπιπτόντως, μια καλή πρακτική είναι να τοποθετούμε το εικονίδιό μας στον φάκελο του Project. Εάν μετά κι από αυτή τη ρύθμιση δοκιμάσουμε να μεταγλωττίσουμε το πρόγραμμά μας, με μεγάλη χαρά θα διαπιστώσουμε ότι το εκτελέσιμο αρχείο που παρήχθη έχει το δικό μας μοναδικό εικονίδιο! Ωστόσο, αν δοκιμάσουμε να τρέξουμε το πρόγραμμα, τόσο στη μπάρα με τον τίτλο της φόρμας όσο και στη γραμμή εργασιών του λειτουργικού θα εμφανίζεται το προεπιλεγμένο εικονίδιο. Για την εμφάνιση του δικού μας εικονιδίου σε αυτές τις θέσεις απαιτείται μία ακόμα ρύθμιση. Συγκεκριμένα, πρέπει να μεταβούμε στις ιδιότητες της φόρμας του προγράμματός μας (όχι στις ιδιότητες ολόκληρου του project) και να δηλώσουμε το εικονίδιό μας στην ιδιότητα Icon. Αυτό ήταν όλο!



Για να εμφανίζεται το εικονίδιο που φτιάξαμε στη γραμμή τίτλου της φόρμας και στη γραμμή εργασιών, πρέπει να το δηλώσουμε και στις ιδιότητες της φόρμας. Ίσως αυτό να μοιάζει ανόητο, αλλά έτσι μπορούμε να έχουμε ξεχωριστά εικονίδια για ξεχωριστές φόρμες, αν η εφαρμογή μας διαθέτει πάνω από μια.

Για να ολοκληρώσουμε τη συζήτηση γύρω από το εικονίδιο, πρέπει να αναφερθούμε σε ένα ακόμα ζήτημα. Δεν θα ήταν ωραία αν το εικονίδιο που σχεδιάσαμε με τόσο μεράκι εμφανιζόταν και στο παράθυρο "About" του deltaPad; Ένας εύκολος τρόπος για να υλοποιήσουμε αυτό το παράθυρο είναι με τη βοήθεια ενός MessageBox. Ωστόσο, το .NET Framework δεν προσφέρει κάποιον άμεσο τρόπο για να επιβάλλουμε την εμφάνιση ενός εικονιδίου καταμεσής ενός MessageBox. Θα μπορούσαμε να πετύχουμε κάτι τέτοιο επεκτείνοντάς την κλάση MessageBox, αλλά αυτό θα αποτελούσε πολύ μεγάλο μπελά για το τίποτα. Έτσι, αποφασίσαμε να υλοποιήσουμε το παράθυρο About δημιουργώντας μια νέα φόρμα. Πιστεύουμε ότι ο τρόπος δημιουργίας και σχεδίασης μιας φόρμας είναι γνωστός και δεν χρειάζεται κάποια περαιτέρω επεξήγηση. Σημειώστε μόνο ότι για την εμφάνιση της εικόνας χρησιμοποιήσαμε ένα στοιχείο PictureBox, για την εμφάνιση των πληροφοριών αξιοποιήσαμε τρία TextBoxes, ενώ για το κλείσιμο του παραθύρου (κλήση της μεθόδου close της ίδιας της φόρμας) χρησιμοποιήσαμε ένα Button. Επιπρόσθετα, φροντίσαμε να τροποποιήσουμε κατάλληλα τις ιδιότητες της φόρμας, ώστε να μην εμφανίζεται στη γραμμή εργασιών του συστήματος, να μην είναι δυνατή η αλλαγή του μεγέθους της και τα κουμπιά μεγιστοποίησης και ελαχιστοποίησης να είναι ανενεργά.

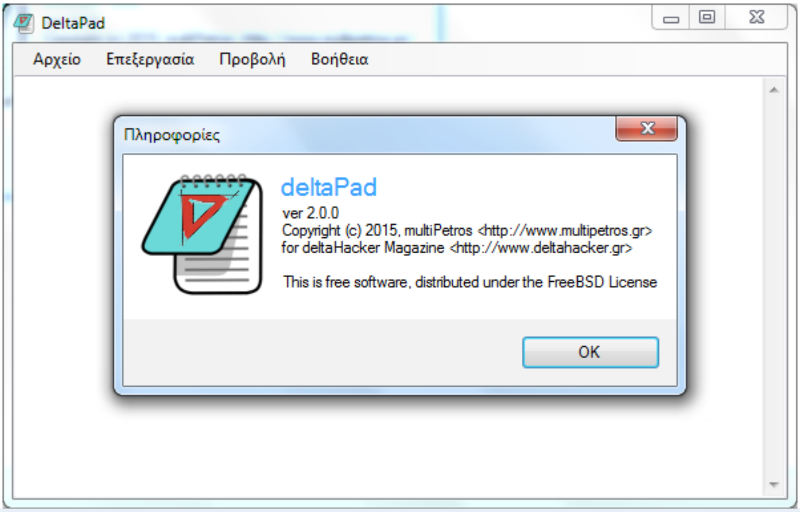


Ο ορισμός του εικονιδίου για το εκτελέσιμο αρχείο γίνεται από την σχετική καταχώρηση, στις ιδιότητες ολόκληρου του Project.

Τέλος, για να εμφανίζεται το δικό μας "About Box" πρέπει να τροποποιήσουμε το κώδικα της μεθόδου που χειρίζεται το συμβάν AboutToolStripMenuItemClick. Η επέμβαση που κάναμε έχει ως εξής:

```
void AboutToolStripMenuItemClick(object sender, EventArgs e) {
    Form about = new AboutForm();
    about.ShowDialog(this);
}
```

Όπως βλέπετε, δημιουργήσαμε ένα στιγμιότυπο της κλάσης AboutForm και καλέσαμε τη μέθοδο ShowDialog, ορίζοντας ως γονικό στοιχείο την κεντρική φόρμα. Η διαφορά της ShowDialog() από την Show() είναι ότι η εμφανιζόμενη φόρμα έχει την συμπεριφορά ενός πλαισίου διαλόγου και δεν επιτρέπει την μετάβασή μας στην γονική φόρμα, αν δεν κλείσουμε πρώτα το πλαίσιο διαλόγου. Σας προτρέπουμε να δοκιμάσετε και τις δύο μεθόδους για να διαπιστώσετε κι εσείς τη διαφορά. Σημειώστε ότι μπορείτε να μελετήσετε ολόκληρο τον κώδικα του ανανεωμένου deltaPad, κατεβάζοντας το σχετικό πακετάκι από το http://bit.ly/dH047dotNETp9.



Το παράθυρο πληροφοριών της εφαρμογής εμφανίζει πλέον το εικονίδιό της, ενώ και ο τίτλος έχει επισημανθεί με μεγαλύτερη γραμματοσειρά και διαφορετικό χρώμα.

Κάπου εδώ θα σας αφήσουμε να απολαύσετε τον δικό μας text editor. Μέχρι το επόμενο τεύχος σας ευχόμαστε πολλά και δημιουργικά απογεύματα, μπροστά από το SharpDevlop και τη C#!

Skill: Intermediate
Tags: NetworkManager, Dnsmasq, upstream server, pfSense, Unbound

Στρατηγικές εκμετάλλευσης του name server μας

Στο άρθρο περί δημιουργίας του δικού μας caching name server, που ξεκινά στη σελίδα 24, φτάσαμε στο σημείο όπου η υπηρεσία είναι έτοιμη και λειτουργεί. Κάναμε και μερικές δοκιμές, διαπιστώνοντας τα οφέλη μιας τοπικής cache. Είναι ώρα να συζητήσουμε τους διάφορους τρόπους κατά τους οποίους τα μηχανήματα του τοπικού μας δικτύου ενημερώνονται για την παρουσία του νέου name server – και φυσικά αρχίζουν να τον εκμεταλλεύονται. Ξέρετε, με την καλή έννοια.

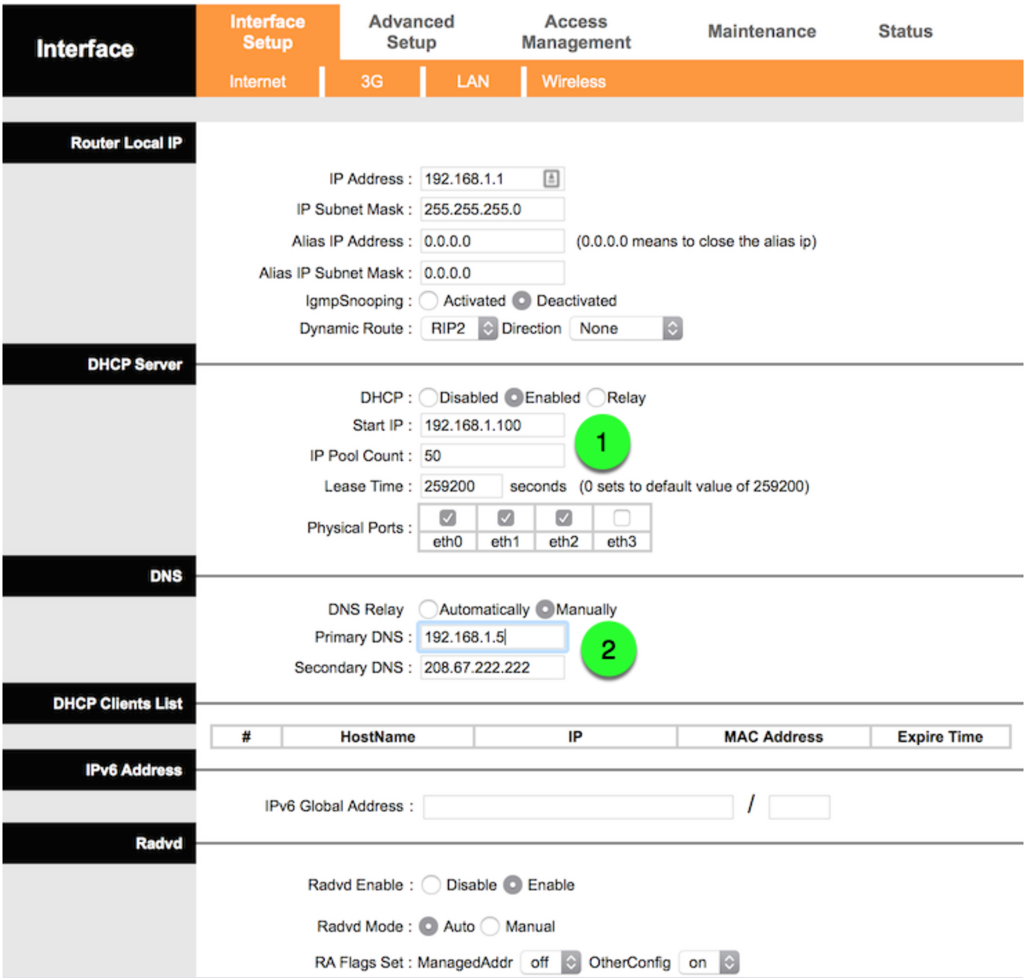
του Χρήστου Βαρελά

Αν χρησιμοποιείτε τον ενσωματωμένο resolver του DSL router σας, παραμερίζοντάς τον για χάρη του νέου που μόλις φτιάξαμε είναι σχεδόν βέβαιο ότι οι δικτυακές επιδόσεις των συσκευών σας θα βελτιωθούν. Ακόμη κι αν έχετε κάποιον ικανό name server στο τοπικό δίκτυο, όπως, π.χ., είναι το Unbound του pfSense, πάντα θα υπάρχουν σενάρια κατά τα οποία ένα ή περισσότερα μηχανήματα του LAN θα έχει νόημα να εκμεταλλεύονται τον καινούργιο name server. Γενικά, αυτό που μας ενδιαφέρει στο παρόν άρθρο είναι οι διάφοροι τρόποι κατά τους οποίους ένα ή περισσότερα ή ακόμη και όλα τα μηχανήματα ενός τοπικού δικτύου αλλάζουν name server για χάρη εκείνου που φτιάξαμε.

Αυτόματη ενημέρωση, από το κέντρο

Εκτός ελαχίστων εξαιρέσεων, το πιθανότερο είναι πως στο τοπικό σας δίκτυο υπάρχει τουλάχιστον ένας DHCP server, π.χ., αυτός του DSL router ή του wireless access point, ώστε οι παράμετροι δικτύωσης (IP, subnet mask, gateway κ.ά.) των clients να παίρνουν τιμές αυτόματα. Από τη στιγμή λοιπόν που έχουμε έτοιμο τον name server μας και λειτουργεί κανονικά, ίσως θελήσουμε να ρυθμίσουμε και τον router του τοπικού δικτύου ώστε να ενημερώνει τους πελάτες για την παρουσία του server. Έως αυτή τη στιγμή γινόταν χρήση των name servers του ISP ή εκείνων κάποιου άλλου provider (π.χ., OpenDNS, Google). Αυτή η κατάσταση είναι ώρα ν' αλλάξει. Τώρα, το πώς ακριβώς ρυθμίζεται ο router διαφέρει από μάρκα σε μάρκα – ακόμη κι από μοντέλο σε μοντέλο της ίδιας μάρκας. Η γενική ιδέα είναι ότι συνδε-

όμαστε στο web panel του router, μεταβαίνουμε στη σελίδα που αφορά στο LAN και κάπου εκεί ζητάμε να μη λαμβάνονται πληροφορίες DNS από τον ISP αλλά στους πελάτες να γνωστοποιείται ο name server που εμείς υποδεικνύουμε.



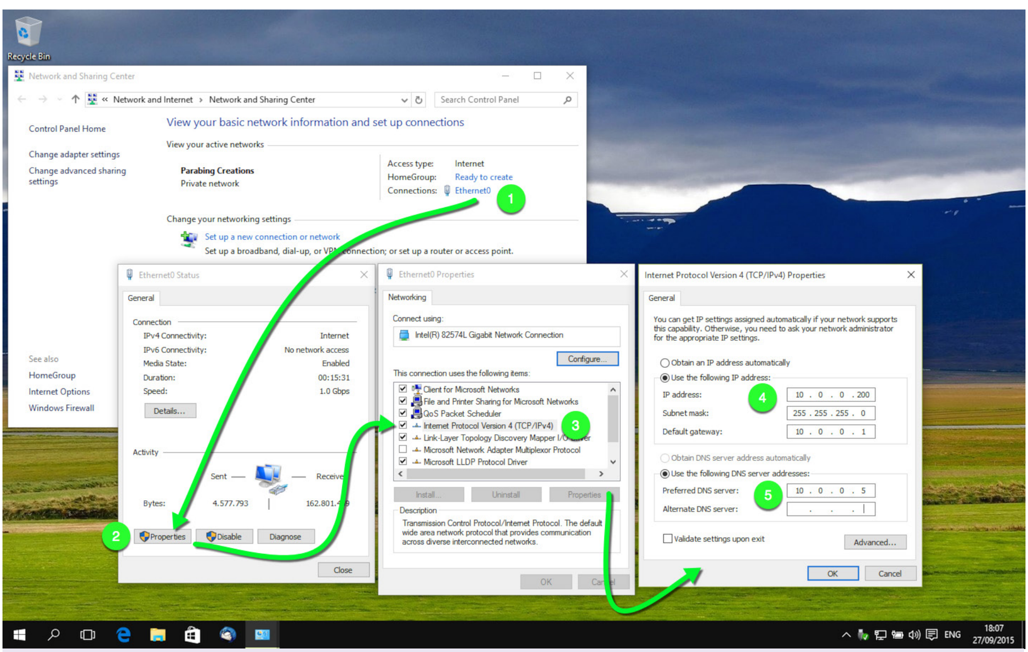
Τροποποίηση της συμπεριφοράς ενός τυπικού ADSL router. Στους πελάτες μοιράζονται αυτόματα διευθύνσεις IP μεταξύ των 192.168.1.100 και 192.168.1.149 συμπεριλαμβανομένων (1). Ο συγκεκριμένος router αρχικά χρησιμοποιούσε τους name servers που έπαιρνε από τον ISP, γυρίσαμε όμως τη σχετική παράμετρο στο "Manually" (2) και πληκτρολογήσαμε τις διευθύνσεις IP του δικού μας name server (192.168.1.5, primary), καθώς κι ενός της OpenDNS (208.67.222.222, secondary). Αν παρ' ελπίδα ο name server μας πέσει (λέμε τώρα), τότε αυτομάτως θα επιστρατευτεί ο άλλος.

Χειροκίνητη ενημέρωση

Όλα –ή έστω κάποια– από τα μηχανήματα ενός LAN είναι πιθανό να *μην* παίρνουν τις δικτυακές παραμέτρους από τον DHCP server, αλλά να διαθέτουν το λεγόμενο static networking configuration. Θεωρούμε ότι, από τη στιγμή που γι' αυτό το configuration υπεύθυνοι είμαστε εμείς οι ίδιοι, δεν θα είναι καθόλου δύσκολο να αλλάξουμε τις ρυθμίσεις DNS ώστε να λαμβάνεται υπόψη ο καινούργιος name server.

Να σημειώσουμε πάντως ότι η σύγχρονη τάση θέλει όλους τους clients μέσα σ' ένα LAN να παίρνουν τις ρυθμίσεις δικτύωσης δυναμικά και κεντρικά, από τον DHCP server (π.χ., του router). Δυναμικό configuration χρησιμοποιούν ακόμη και οι συσκευές που για τον οποιονδήποτε λόγο χρειάζονται στατική διεύθυνση IP. Απλά τα εν λόγω μηχανήματα παίρνουν πάντα το ίδιο IP, με βάση το MAC address της ασύρματης ή της ενσύρματης κάρτας δικτύου μέσω της οποίας συμμετέχουν στο δίκτυο. Σε ένα τέτοιο setup διαφορετικά μηχανήματα είναι πιθανόν να μαθαίνουν για διαφορετικούς name servers. Περισσότερα επ' αυτού συζητάμε στην τελευταία ενότητα του παρόντος άρθρου.

Κάποια μηχανήματα εξάλλου ίσως παίρνουν όλες τις παραμέτρους δικτύωσης δυναμικά, εκτός από τον name server τους που έχει οριστεί χειροκίνητα. Ως ένα παράδειγμα δείτε το ακόλουθο screenshot, στο οποίο φροντίζουμε ώστε ένα μηχάνημα με Windows 10 να χρησιμοποιεί τον name server μας.



Παράδειγμα τροποποίησης των παραμέτρων δικτύωσης ενός μηχανήματος με Windows 10, ώστε να χρησιμοποιεί τον ολοκαίνουργιο name server με IP το 10.0.0.5. Παρατηρήστε ότι οι παράμετροι IP address, Subnet mask και Default gateway είναι δυνατόν να παίρνουν τιμές αυτόματα (από τον DHCP server), αλλά το DNS να ορίζεται χειροκίνητα.

Το Linux, ο NetworkManager και το DNS

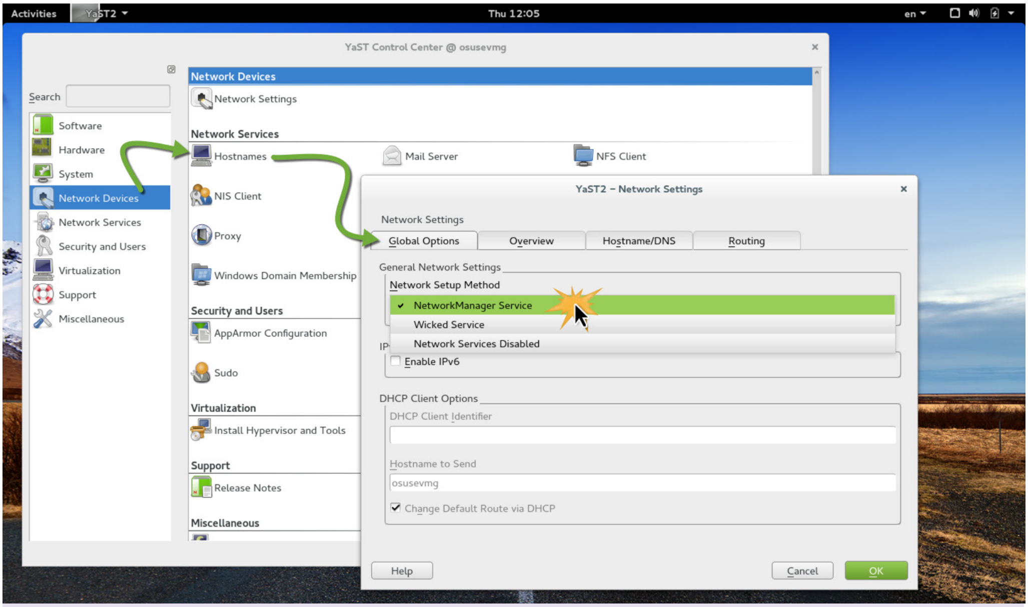
Σε αντίθεση με ό,τι συνέβαινε παλαιότερα με όλες τις διανομές Linux, σε αρκετές σύγχρονες που προρίζονται για χρήση desktop (κι όχι μόνο) τα θέματα δικτύωσης διευθετούνται αυτόματα. Λίγα χρόνια πριν, παραμέτρους όπως διεύθυνση IP, subnet mask, DNS, router κ.ο.κ., στις διανομές Linux τις ορίζαμε μόνοι μας τροποποιώντας ένα ή δύο αρχεία απλού κειμένου. Είχαμε βέβαια και τη δυνατότητα για dynamic configuration, ωστόσο και η ίδια η επιλογή μεταξύ dynamic και static configuration ήταν επίσης δική μας. Σήμερα είναι πράγματι εφικτό (ή πιθανό) να εργαζόμαστε παρομοίως, υπάρχει όμως μια ισχυρή τάση προς τον αυτοματισμό. Έτσι, αν *δεν* σκοπεύουμε να απενεργοποιήσουμε τη σχετική υπηρεσία αυτόματης ρύθμισης δικτύωσης στη διανομή μας, καλό είναι να γνωρίζουμε το πώς λειτουργεί και, κυρίως, πώς τροποποιούμε τη συμπεριφορά της.

Μια δημοφιλής υπηρεσία αυτόματης διευθέτησης θεμάτων δικτύωσης παρέχεται από τον NetworkManager (https://wiki.gnome.org/Projects/NetworkManager). Η ανάπτυξή του ξεκίνησε το 2004 από τη Red Hat (https://www.redhat.com) και κάποια στιγμή βρέθηκε υπό την αιγίδα του GNOME Project (https://www.gnome.org). Ο NetworkManager σχεδιάστηκε ώστε να διευκολύνονται οι κάτοχοι laptop με Linux, καθώς μεταβαίνουν από ασύρματο δίκτυο σε ασύρματο δίκτυο. Χειρίζεται επιπλέον κι ενσύρματες συνδέσεις, τις οποίες μάλιστα προτιμά. Στη σειρά προτίμησης ακολουθούν οι ασύρματες με SSID στο οποίο έχουμε συνδεθεί ξανά κατά το παρελθόν, και μετά οι ασύρματες με SSID στο οποίο *δεν* έχουμε συνδεθεί ξανά κατά το παρελθόν. Αξίζει να υπογραμμίσουμε ότι ο NetworkManager κάνει ό,τι μπορεί ώστε η δικτυακή μας σύνδεση να μη διακόπτεται – ή τουλάχιστον να μην το αντιλαμβανόμαστε: Όχι μόνον όταν αλλάζει το ασύρματο δίκτυο, αλλά ακόμη κι όταν αφήνουμε μια ενσύρματη σύνδεση, π.χ., επειδή βγάλαμε το καλώδιο Ethernet από το laptop αλλά στο χώρο υπάρχει ένα προτιμώμενο ασύρματο δίκτυο.

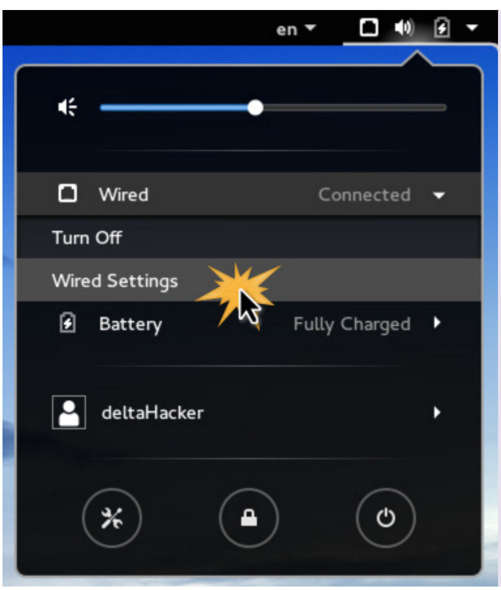
Από αρχιτεκτονικής σκοπιάς, ο NetworkManager αποτελείται από δύο κύρια μέρη:

- την υπηρεσία που χειρίζεται τις ίδιες τις συνδέσεις, τις μεταβάσεις από σύνδεση σε σύνδεση και τις αναφορές περί των αλλαγών
- τα διάφορα front-ends γραφικών που διευκολύνουν το χειρισμό της υπηρεσίας, αν κι όποτε ο χρήστης επιθυμεί να παρεμβαίνει και να τροποποιεί τη λειτουργία της.

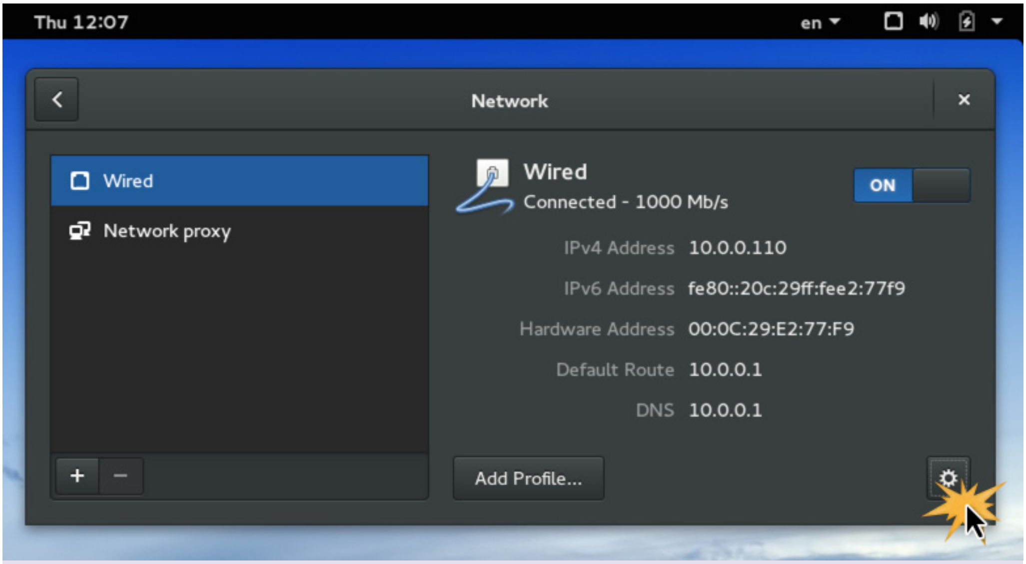
Σε διανομές desktop όπως το Ubuntu, το openSUSE, το elementary OS κ.ά., μαζί με την υπηρεσία του NetworkManager έχουμε και front-ends για το αντίστοιχο περιβάλλον γραφικών. Δείτε στα ακόλουθα screenshots πώς τροποποιούμε τη συμπεριφορά του NetworkManager σε ένα σύστημα Linux με openSUSE 13.2, καθώς και σε ένα με Ubuntu 15.04. Σε κάθε περίπτωση, αυτό που μας ενδιαφέρει είναι να επιστρατεύεται ο δικός μας name server.



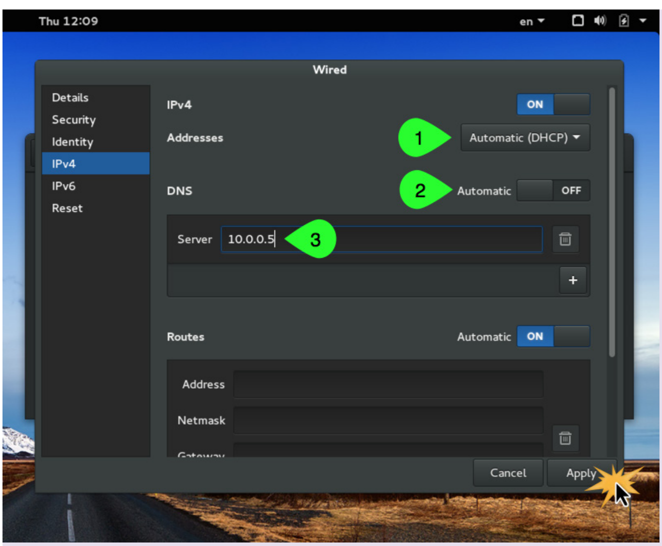
Από το YaST, το πολυεργαλείο διαχείρισης συστήματος του openSUSE, βλέπουμε ότι για τη δικτύωση παρέχονται δύο μέθοδοι αυτόματης διευθέτησης: μέσω του NetworkManager ή μέσω του Wicked Service.



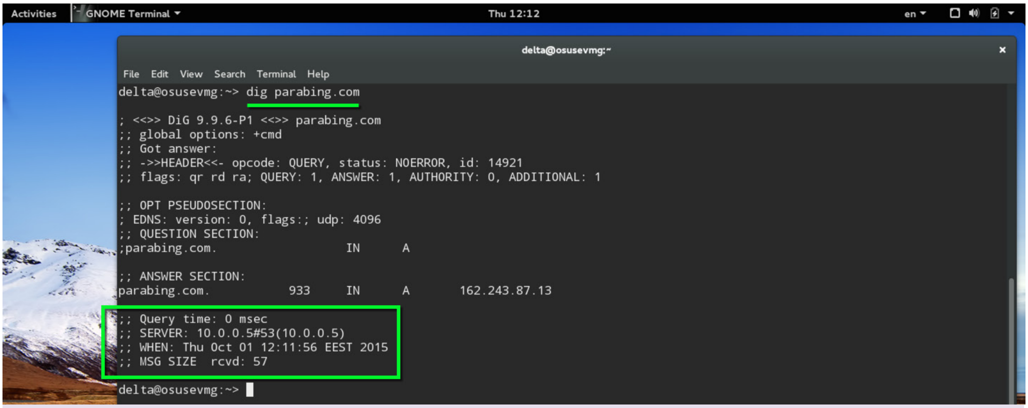
Ο NetworkManager αποδεικνύεται ιδιαίτερα βολικός όταν έχουμε laptop, το οποίο παίρνουμε μαζί μας όπου πηγαίνουμε και συνεπώς συμμετέχει σε πολλά δίκτυα. Επιλέγοντας στο openSUSE το NetworkManager Service, ο έλεγχος της δικτύωσης παύει να γίνεται μέσω του YaST κι αντ' αυτού καταφεύγουμε στο κατάλληλο front-end γραφικών για τον NetworkManager. Στο συγκεκριμένο παράδειγμα τρέχουμε την εκδοχή GNOME του openSUSE και στο front-end του NetworkManager φτάνουμε από την περιοχή με τα εικονίδια ελέγχου, πάνω δεξιά. Εναλλακτικά, αρκεί ένα κλικ στο εικονίδιο Network, του παραθύρου All Settings.



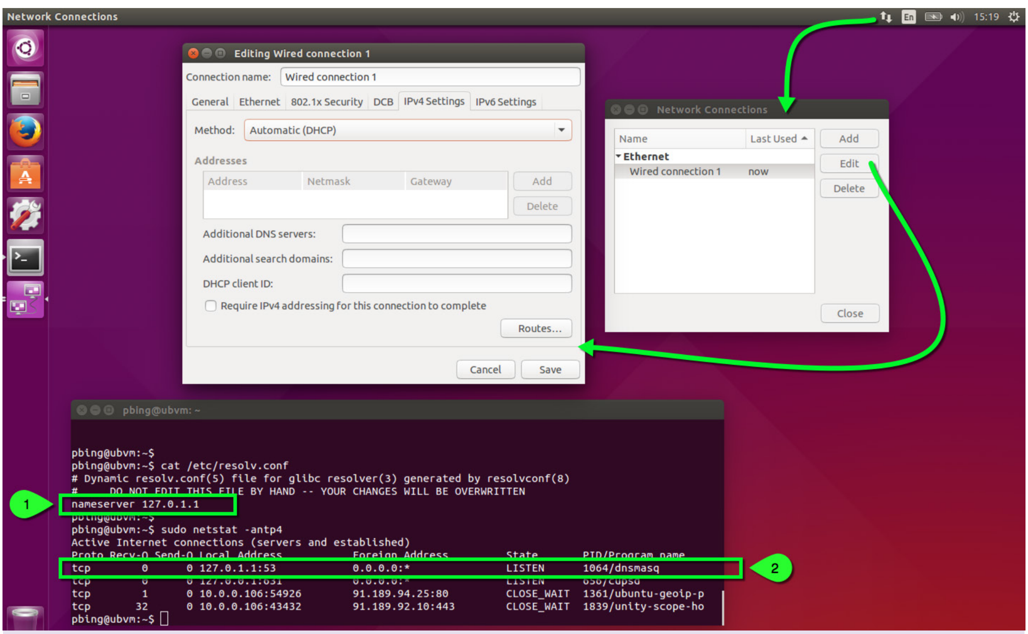
Οι ρυθμίσεις δικτύωσης του wired interface (η μοναδική κάρτα Ethernet του συστήματός μας), όπως φαίνονται μέσα από το applet του NetworkManager. Ο name server που χρησιμοποιείται είναι ο 10.0.0.1 και είναι αυτός του (φθηνιάρικου) router μπροστά από το μηχάνημα. Εμείς φυσικά θέλουμε τον name server που μόλις στήσαμε, ο οποίος βρίσκεται στο Ubuntu Server box, με IP το 10.0.0.5. Κάνουμε λοιπόν την επέμβασή μας, ξεκινώντας με ένα κλικ στο σχετικό εικονίδιο.



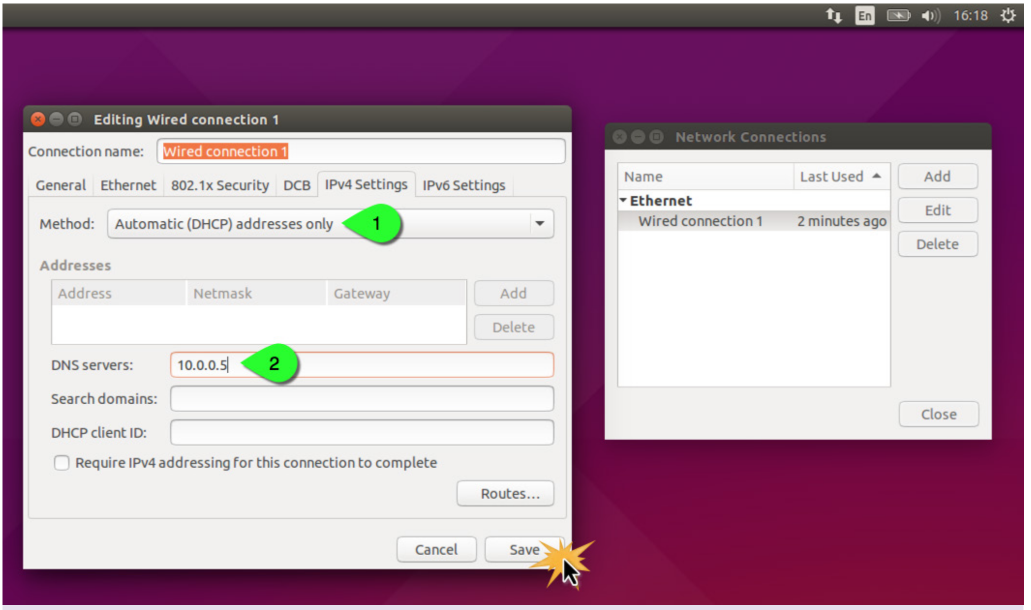
Στις ρυθμίσεις που αφορούν στη δικτύωση IPv4, παρατηρήστε ότι αφήνουμε τη διευθυνσιοδότηση να γίνεται αυτόματα (Addresses = Automatic (DHCP)). Για το DNS όμως φροντίζουμε ώστε να ισχύει Automatic = OFF. Στη γραμμή ονόματι Server, από κάτω, πληκτρολογούμε τη διεύθυνση IP του επιθυμητού name server (10.0.0.5 για το παράδειγμά μας). Επικυρώνουμε τις αλλαγές με ένα κλικ στο κουμπί Apply, κάτω δεξιά. Προκειμένου να βεβαιωθούμε ότι οι αλλαγές έχουν ληφθεί υπόψη, αρκεί να επανεκκινήσουμε τον NetworkManager. Για παράδειγμα, σε ένα παράθυρο τερματικού δίνουμε "sudo service NetworkManager restart" (χωρίς τα εισαγωγικά).



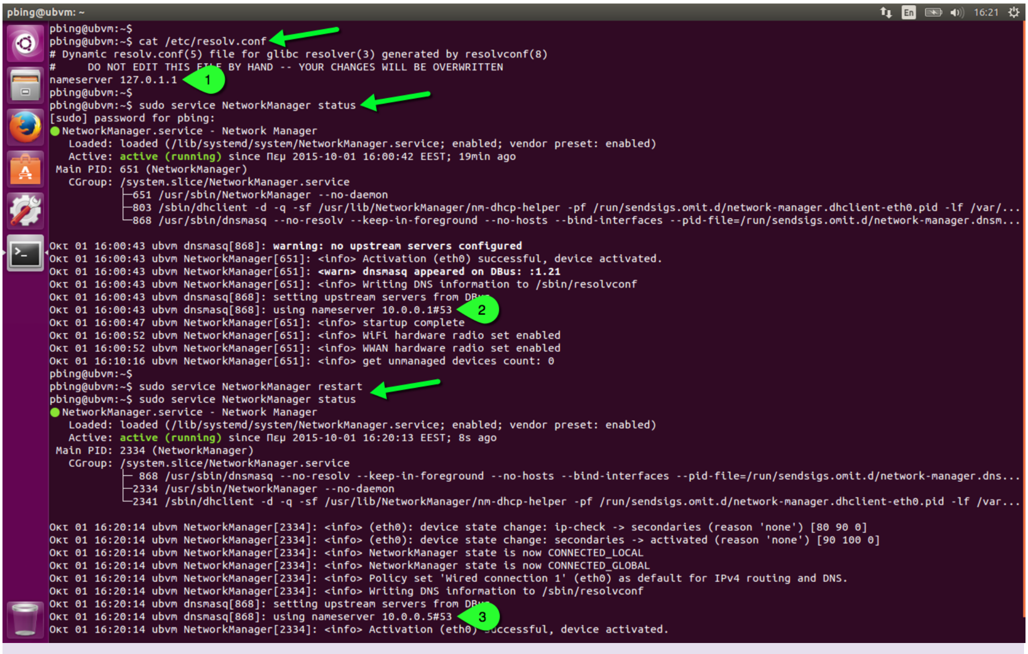
Το ωραιότατο openSUSE μας χρησιμοποιεί τον ωραιότατο name server μας, στο 10.0.0.5. Μερικές δοκιμές με το dig το επιβεβαιώνουν.



Ανοίγοντας το applet του NetworkManager στο Ubuntu, βλέπουμε ότι εξ ορισμού η δικτύωση ρυθμίζεται αυτόματα (Method = automatic (DHCP)). Με μια ματιά στο αρχείο /etc/resolv.conf, εξάλλου, διαπιστώνουμε ότι δεν επιστρατεύεται κάποιος name server από το τοπικό δίκτυο – ούτε καν αυτός του router (1). Αντίθετα, φαίνεται ότι στο σύστημα τρέχει ένας name server κι ο NetworkManager φροντίζει ώστε αυτός να είναι ο προκαθορισμένος. Παρατηρώντας την έξοδο του netstat επιβεβαιώνουμε την υποψία μας: ο τοπικός name server είναι το Dnsmasq (http://www.thekelleys.org.uk/dnsmasq/doc.html) (2). Πρόκειται για έναν μικρό κι ελαφρύ forwarder κι αυτή η προσέγγιση του Ubuntu είναι εξαιρετικά βολική, υπό την έννοια ότι το σύστημα διαθέτει όσο πιο κοντά του γίνεται έναν ικανότατο caching name server.



Παρά το γεγονός ότι το Ubuntu χρησιμοποιεί το τοπικό Dnsmasq, ο δικός μας name server μπορεί, αν φυσικά το θέλουμε, να μπει κι εκείνος στο παιχνίδι. Για την ακρίβεια, σε περίπτωση που δεν εργαζόμαστε σε κάποιο laptop αλλά σ' έναν υπολογιστή που μένει στο ίδιο δίκτυο, τότε ο name server μας είναι δυνατόν να ‹ναι αυτός στον οποίο θα προωθεί τα αιτήματα λειτουργικού κι εφαρμογών το Dnsmasq. Προς τούτο, ανοίγουμε το applet του NetworkManager και τροποποιούμε τις ρυθμίσεις του network interface που μας ενδιαφέρει (στο παράδειγμά μας είναι ένα Ethernet interface κι ονομάζεται "Wired connection 1"). Στην καρτέλα IPv4 Settings του νέου παραθύρου που εμφανίζεται, αν η διευθυνσιοδότηση θέλουμε να συνεχίσει να 'ναι αυτοματοποιημένη τότε φροντίζουμε ώστε να ισχύει Method = Automatic (DHCP) addresses only (1). Στη θυρίδα DNS servers, όμως, λίγο πιο κάτω, πληκτρολογούμε τη διεύθυνση IP του επιθυμητού name server (2). Επικυρώνουμε την αλλαγή με ένα κλικ στο κουμπάκι Save, κάτω δεξιά στο παράθυρο.



Μετά τη μικροαλλαγή στη συμπεριφορά του NetworkManager, ανοίγουμε ένα παράθυρο τερματικού και κάνουμε μερικές δοκιμές. Η αλλαγή περί DNS που μόλις πραγματοποιήσαμε δεν αντικατοπτρίζεται στο /etc/resolv.conf, όπως εύκολα διαπιστώνουμε γράφοντας "cat /etc/resolv.conf" (1). Αυτό είναι αναμενόμενο, αφού ο name server που καθορίσαμε είναι αυτός που θα χρησιμοποιεί σε λίγο το Dnsmasq. Ορίσαμε, μ' άλλα λόγια, τον upstream server του Dnsmasq. Πληκτρολογώντας "sudo service NetworkManager status" παρατηρούμε ότι προς το παρόν ο upstream server είναι ο 10.0.0.1 – κι αυτός είναι ο router του τοπικού μας δικτύου (2). Επανεκκινούμε τον NetworkManager με ένα "sudo service NetworkManager restart" κι ελέγχουμε και πάλι τι ισχύει, δίνοντας ξανά "sudo service NetworkManager status". Αυτή τη φορά βλέπουμε ότι ο upstream server είναι ο 10.0.0.5 – κι αυτός είναι ο name server μας (3).

Το Ubuntu Server κι ο δικός του name server

Το Ubuntu Server box, στο οποίο εργαστήκαμε στο άρθρο της σελίδας PQYXYZ κι εγκαταστήσαμε το BIND, λαμβάνει αυτομάτως τις παραμέτρους δικτύωσης από τον DHCP του τοπικού δικτύου. Αυτό σημαίνει ότι ο name server που χρησιμοποιεί για το resolving υποδεικνύεται "εξωτερικά» και συνεπώς παρακάμπτεται η τοπική εγκατάσταση του BIND. Δείτε, π.χ., τι ισχύει στο μηχάνημα των δοκιμών μας:

```
~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.0.1
```

Το σύστημα και οι εφαρμογές εξυπηρετούνται από τον 10.0.0.1, ο οποίος υπεδείχθη από τον DHCP του δικτύου. Αν θέλουμε να εξυπηρετούνται από το BIND, τότε *πριν* από τον προαναφερθέντα name server πρέπει να βρίσκεται ο 127.0.0.1. Βλέπουμε πάντως ότι το αρχείο /etc/resolv.conf δεν έχει νόημα να το τροποποιήσουμε με το χέρι, αφού παράγεται δυναμικά και κατά το επόμενο boot θα αλλάξει, συνεπώς η όποια τροποποίηση έχουμε κάνει θα χαθεί. Η δυναμική παραγωγή του /etc/resolv.conf είναι ευθύνη της υπηρεσίας ονόματι resolvconf και για τη συγκρότησή του χρησιμοποιούνται τα αρχεία απλού κειμένου head και base, μέσα στον κατάλογο /etc/resolvconf/resolv.conf.d. Δείτε τα περιεχόμενα του head:

```
~$ cat /etc/resolvconf/resolv.conf.d/head
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
```

Περιέχει το ψαρωτικό μήνυμα που είδαμε προηγουμένως κι αποτελεί την αρχή του /etc/resolv.conf. Ε, αν στο τέλος του head προσθέσουμε το "nameserver 127.0.0.1", τότε ανεξαρτήτως του name server που υποδεικνύεται εξωτερικά ο *πρώτος* που θα χρησιμοποιείται από σύστημα κι εφαρμογές θα είναι ο 127.0.0.1, δηλαδή η τοπική εγκατάσταση του BIND. Ας προσθέσουμε τη γραμμή:

```
~$ sudo su
/home/admin# echo "nameserver 127.0.0.1" >> /etc/resolvconf/resolv.conf.d/head
/home/admin# exit
exit
~$
```

Ωραία. Στο /etc/resolv.conf δεν έχουν περάσει οι αλλαγές και για να συμβεί κι αυτό αρκεί να επανεκκινήσουμε την υπηρεσία resolvconf:

```
~$ sudo service resolvconf restart
resolvconf stop/waiting
resolvconf start/running
~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
nameserver 10.0.0.1
~$
```

Πάρα πολύ ωραία. Ας κάνουμε και μια δοκιμή με το dig:

```
~$ dig isafjordur.is
; <<>> DiG 9.9.5-3ubuntu0.5-Ubuntu <<>> isafjordur.is
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 59452
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;isafjordur.is.       IN A
;; Query time: 22 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Oct 03 14:57:06 EEST 2015
;; MSG SIZE  rcvd: 42
```

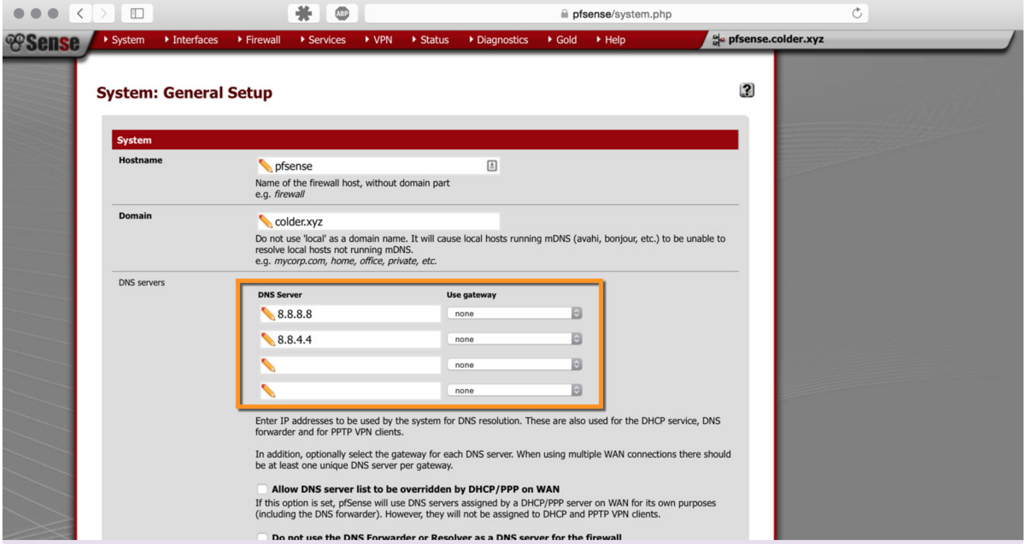
Απαντά ο 127.0.0.1 — κι αυτό είναι ό,τι ακριβώς θέλουμε.

Το pfSense, οι name servers του κι ο DHCP

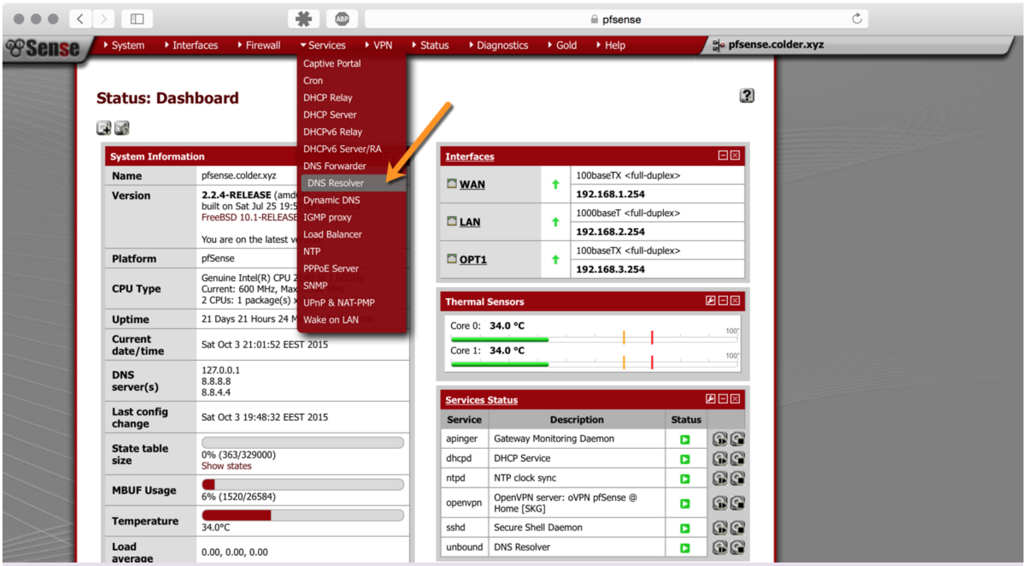
Το pfSense (https://www.pfsense.org) είναι ένα από τα αγαπημένα μας λειτουργικά συστήματα για χρήση router/firewall. Σε περίπτωση που δεν το γνωρίζετε ή δεν έχετε βρει χρόνο να το δοκιμάσετε, παρακολουθήστε το 7ο επεισόδιο της 1ης σεζόν του deltaCast στο http://deltahacker.gr/deltacast-s01e07. Ο name server που χρησιμοποιούσε το pfSense έως την έκδοση 2.2 ήταν το Dnsmasq και στο μενού Services του web panel αναγραφόταν ως "DNS Forwarder". Από την έκδοση 2.2 κι έπειτα υπάρχει κι άλλος ένας resolver στο pfSense: Πρόκειται για το Unbound, στο μενού Services αναφέρεται ως "DNS Resolver", και στις νέες εγκαταστάσεις του λειτουργικού είναι ενεργοποιημένος εξ ορισμού. (Το Dnsmasq εξακολουθεί να είναι παρόν, αλλά δεν είναι ενεργό.)

Το Unbound έχει τη δυνατότητα να λειτουργεί είτε ως recursive είτε ως forwarding name server. Είναι περισσότερο από ικανό για να εξυπηρετεί το τοπικό σας δίκτυο, όσοι κι αν είναι οι ενεργοί πελάτες. Πάντως σε κάποιες περιπτώσεις είναι πιθανό να θέλετε κάποιον ξεχωριστό name server για ένα σύνολο μηχανημάτων – κι αυτός ο ξεχωριστός name server κάλλιστα μπορεί να είναι εκείνο το Ubuntu Server box με το BIND. Χάρη στην ευελιξία του DHCP server που διαθέτει το pfSense, ο καθορισμός name server γίνεται είτε κατά πελάτη είτε κατά network interface. Σίγουρα για ένα μικρό δίκτυο δεν έχει νόημα να χρησιμοποιείτε πάνω από έναν τοπικό name server, εκτός βέβαια κι αν το κάνετε για εκπαιδευτικούς λόγους. Το pfSense όμως είναι σε θέση να εξυπηρετεί περισσότερα από ένα δίκτυα. Ίσως, π.χ., στο LAN interface να βρίσκονται μόνο wired clients, ενώ στο OPT1 μόνο wireless clients κι αυτοί να θέλετε να εξυπηρετούνται μέσω άλλου name server.

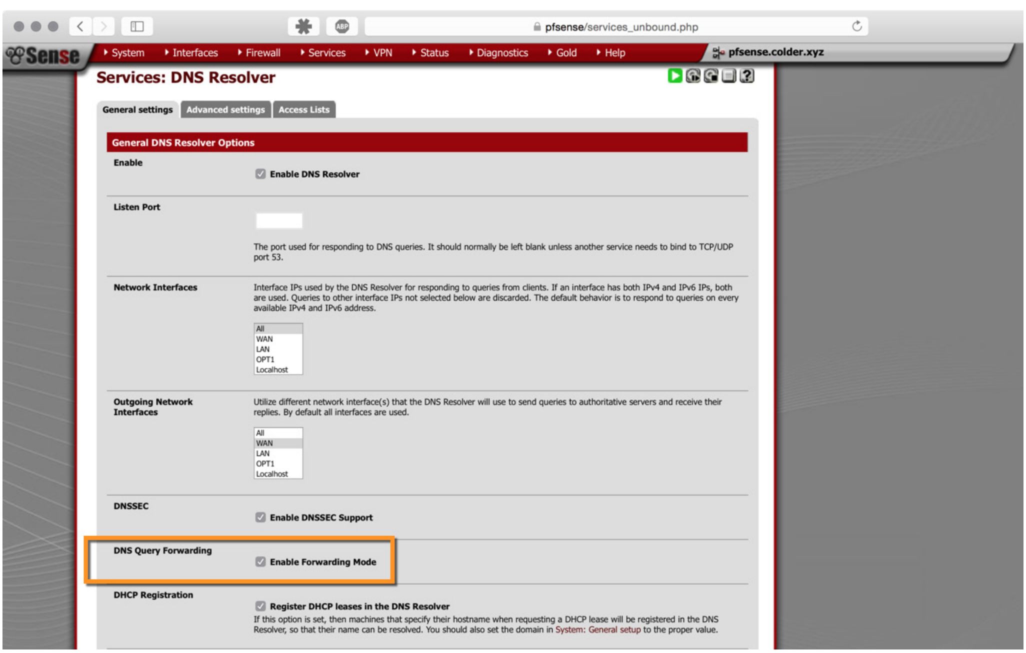
Υπάρχουν κι άλλες δυνατότητες, η ανάλυση των οποίων ξεφεύγει από τους σκοπούς του παρόντος. Θα σημειώσουμε μόνο ότι το pfSnese παρέχει μεγάλη ευελιξία κι αξίζει να το γνωρίσετε. Αν όχι για χρήση στο σπίτι, τότε σίγουρα για το εργαστήριο στο σχολείο, για το χώρο εργασίας κ.ο.κ. Δείτε τα ακόλουθα screenshots και διαβάστε τις αντίστοιχες περιγραφές. Ελπίζουμε να εμπνευστείτε λίγο περισσότερο και, γιατί όχι, ν' αλλάξετε εντελώς τα χαρακτηριστικά των δικτύων για τα οποία είστε υπεύθυνος.



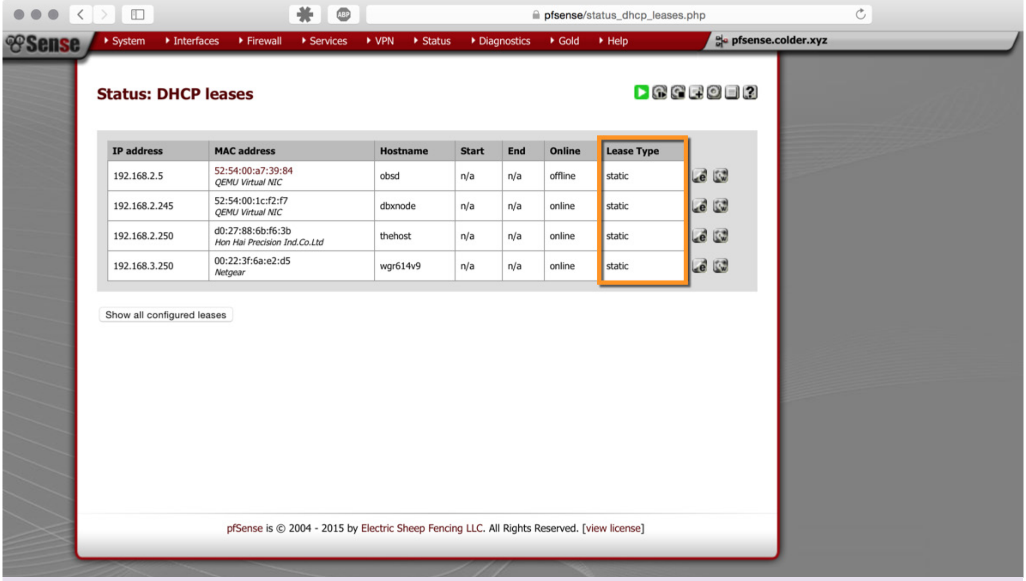
Στις γενικές ρυθμίσεις του pfSense πρέπει να υπάρχει η διεύθυνση IP τουλάχιστον ενός name server – και στο παράδειγμα υπάρχουν οι διευθύνσεις IP των δύο δημοσίων name servers της Google. Τώρα, καθένας από τους δύο προεγκατεστημένους στο pfSense name servers είναι δυνατόν να χρησιμοποιεί, όταν φυσικά είναι ενεργοποιημένος, εκείνους που έχουμε πληκτρολογήσει εδώ. Ακριβέστερα, ο DNS Forwarder (πρόκειται για το Dnsmasq) τους χρησιμοποιεί εξ ορισμού, ενώ ο DNS Resolver (πρόκειται για το Unbound) τους χρησιμοποιεί μόνον όταν λειτουργεί ως απλός forwarder (κι όχι ως resolver).



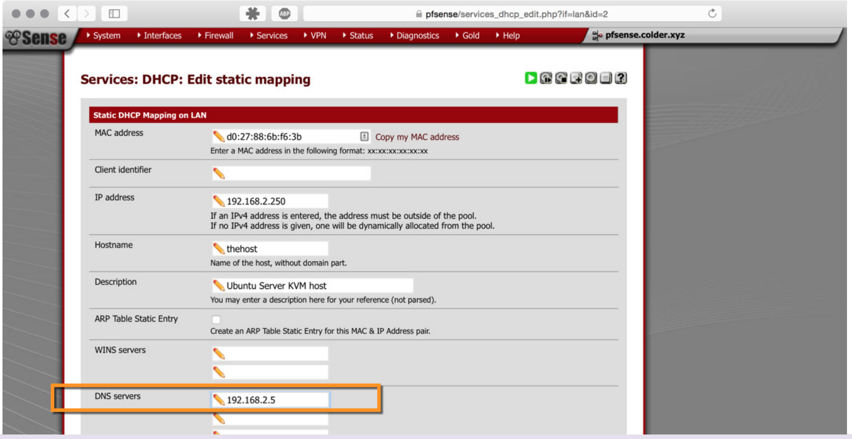
Οι δύο διαθέσιμοι name servers του pfSense, στο μενού Services: έναν από τους δύο χρησιμοποιούν οι clients του router, εκτός κι αν για κάποια μηχανήματα ή ολόκληρα δίκτυα έχουμε ορίσει εξαιρέσεις. Πριν την έκδοση 2.2 του pfSense παρών ήταν μόνον ο DNS Forwarder (το Dnsmasq). Από την έκδοση 2.2 και μετά υπάρχει κι ο DNS Resolver (το Unbound). Στις νέες εγκαταστάσεις του pfSense ενεργοποιημένος είναι ο DNS Resolver.



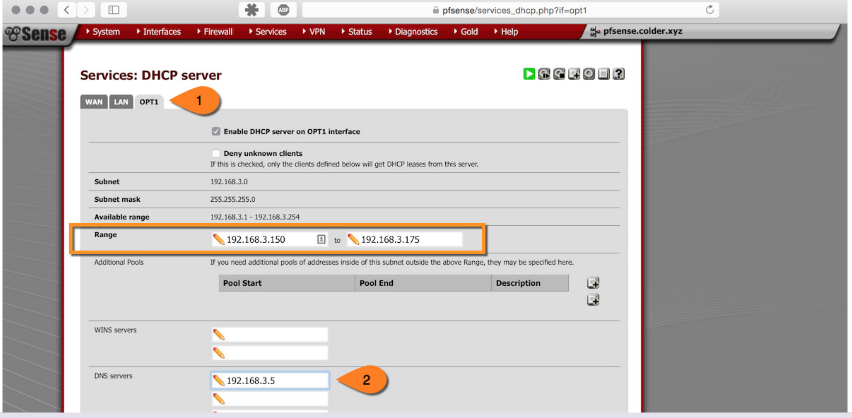
Η προκαθορισμένη συμπεριφορά στον DNS Resolver (Unbound) είναι να τιμά τ' όνομά του, δηλαδή να λειτουργεί ως resolver. Αν όμως θέλουμε να ελαφρύνουμε κάπως τον φόρτο εργασίας του router, τότε κάλλιστα μπορούμε να ζητήσουμε από το Unbound να λειτουργεί ως forwarder. Σ' αυτή την περίπτωση η υποστήριξη DNSSEC από πλευράς Unbound πρέπει να ‹ναι ενεργοποιημένη και οι upstream DNS servers επίσης να υποστηρίζουν το DNSSEC (συμβαίνει με τους public DNS servers της Google). Περιττό επίσης να υπογραμμίσουμε ότι οι upstream DNS servers θα πρέπει να είναι της εμπιστοσύνης μας. Εμείς, π.χ., χωρίς να έχουμε βάσιμους λόγους αλλά κυρίως ως θέμα αρχής, δεν εμπιστευόμαστε τους name servers των ISPs. Με εκείνους της Google όμως αισθανόμαστε πιο άνετα. Τέλος, έχετε κατά νου πως η λειτουργία του Unbound ως forwarder είναι υποχρεωτική για όσους βγαίνουν στο Internet από δύο ή περισσότερες διαφορετικές συνδέσεις. Αναφερόμαστε στο λεγόμενο multi-WAN που υποστηρίζει το pfSense, σε load balancing ή σε failover, κι αν δεν έχετε ιδέα περί τίνος πρόκειται τότε προτείνουμε να διαβάσετε το συντομότερο δυνατόν το σχετικό άρθρο (http://deltahacker.gr/?p=10885), στο τεύχος 029 του περιοδικού (http://deltahacker.gr/deltahacker029).



Ιδού οι clients που αυτή τη στιγμή είναι συνδεδεμένοι *απευθείας* στο pfSense μας (μέσω των LAN και OPT1 interfaces). Όλοι τους υιοθετούν το dynamic networking configuration, γεγονός που σημαίνει ότι τις τιμές για τις βασικές παραμέτρους δικτύωσης (IP address, router address, DNS κ.ά.) τις λαμβάνουν από τον διαθέσιμο DHCP server. Για τους τέσσερις εικονιζόμενους clients έχουμε ζητήσει από τον DHCP του pfSense να τους δίνει πάντα την ίδια διεύθυνση IP (τους clients τους ξεχωρίζει με βάση το MAC address). Γι' αυτό και στη στήλη Lease Type αναγράφεται παντού αυτό το "static". Ως προς τη διαμόρφωση των παραμέτρων δικτύωσης των clients, ο DHCP του pfSense παρέχει ακόμα μεγαλύτερη ευελιξία. Διαβάστε παρακάτω.



Παράδειγμα της ευελιξίας που παρέχει ο DHCP server του pfSense: Ειδικά για τον client με διεύθυνση IP την 192.168.2.250 (και hostname αυτό το κάπως μυστηριώδες "thehost"), δεν χρησιμοποιείται κάποιος από τους name servers του συστήματος (Dnsmasq και Unbound) αλλά εκείνος στο 192.168.2.5.



Ο DHCP server του pfSense box είναι ενεργοποιημένος για τα LAN και OPT1 network interfaces κι εξυπηρετεί τους πελάτες που συνδέονται σε καθένα εξ αυτών. Στο screenshot βλέπουμε το OPT1 interface (1). Ο DHCP μοιράζει δυναμικά διευθεύνσεις IP μεταξύ 192.168.3.150 και 192.168.3.175 συμπεριλαμβανομένων. Στα μηχανήματα πάνω στο OPT1, τα οποία χρησιμοποιούν dynamic networking configuration και συνεπώς εξυπηρετούνται από τον DHCP, ο name server που τους γνωστοποιείται είναι ο 192.168.3.5 κι όχι κάποιος από εκείνους που διαθέτει το pfSense (Dnsmasq και Unbound). Γιατί θα θέλαμε μια τέτοια διευθέτηση; Για πολλούς λόγους, αλλά προς το παρόν θα αναφέρουμε μόνον έναν: Ο name server στο 192.168.3.5 θα μπορούσε να κάνει το resolving μέσω του δικτύου ανωνυμίας Tor, επομένως τα DNS requests των πελατών πίσω από το OPT1 θα ήταν, πρακτικά, ανώνυμα. Παρατηρήστε τέλος και τη θυρίδα Gateway, λίγο πιο κάτω: Με την κατάλληλη προεργασία, οι πελάτες του OPT1 θα ήταν να δυνατόν να βγαίνουν στον έξω κόσμο μέσω του Tor, φυσικά χωρίς να τους έχουμε εγκαταστήσει το κατάλληλο λογισμικό. Περισσότερα επ' αυτού σε επόμενο άρθρο (εκτός κι αν δεν ενδιαφέρεστε, δηλαδή :D)



MORE COLORS AVAILABLE
deltaHacker Field Bag
€25.50
deltaHacker Mugs
€12.50
deltaHacker Thermos® Can Cooler
€17.50
QUESTIONS? | ORDER BY PHONE? | CALL 1-877-809-1659
Where's My Stuff?
Track your recent orders.
View or change your order.
Shipping & Returns
See our shipping rates.
Make a return or exchange.
Need Help?
Satisfaction Guaranteed.
Visit our Customer Service.
is powered by CafePress Inc. Copyright © 1999-2015 CafePress.com. All rights reserved.
olicy | Trademark & Copyright Information
urrency rates are updated daily and may fluctuate.
KER
Δώρα με στιλ, που έχουν *αληθινή* αξία.
Όλα στο μαγαζάκι του deltaHacker,
κάτω από το Cafepress.
Περάστε μια βόλτα από το
http://www.cafepress.co.uk/deltahacker
–σας περιμένουμε!

Skill: Intermediate
Tags: R/2R network, DAC, Color Look-Up Table, true color, transient currents

Κατασκευάστε έναν υπολογιστή από το μηδέν, μέρος 2

Από τη θεωρία στην πλακέτα

Στο παρόν άρθρο εξετάζουμε τα κυκλώματα του υπολογιστή μας με κάθε λεπτομέρεια. Στο τέλος θα έχετε όλα τα απαραίτητα εφόδια για να τον κατασκευάσετε κι εσείς ή, αν είσαστε περισσότερο φιλόδοξοι, να βασιστείτε πάνω του και να φτιάξετε έναν καλύτερο. Σε κάθε περίπτωση, βάλτε το κολλητήρι να ζεσταίνεται κι αρχίστε την αναζήτηση των εξαρτημάτων.

του Παναγιώτη Βαρελά

Ο κόσμος των PC γεννήθηκε με την έλευση του επεξεργαστή 8088, που είχε σχεδιάσει το ισραηλινό παράρτημα της Intel. (Ναι, αυτό το παράρτημα υπήρχε από τότε κι αν δεν σας λέει τίποτα, έχετε υπόψη ότι πρόκειται για την ομάδα που σχεδίασε τις αρχιτεκτονικές Nehalem, Sandy Bridge και Ivy Bridge, ενώ συμμετείχε και στη σχεδίαση του Haswell.) Η μαμά εταιρεία ήθελε έναν επεξεργαστή παρόμοιο με τον 8086, ο οποίος θα μπορούσε να τοποθετηθεί σε φθηνά και πιο εμπορικά μηχανήματα. Ξέρετε πώς κατάφεραν να εξασφαλίσουν το μειωμένο κόστος; Ο 8086 ήταν ένας 16μπιτος επεξεργαστής με 16μπιτο data bus. Αυτό σημαίνει ότι μπορούσε να κάνει πράξεις με αριθμούς των 16 bits, ενώ αντάλλασσε δεδομένα με τα περιφερειακά του μέσα από έναν δίαυλο δεδομένων των 16 γραμμών. Ο σημαντικότερος λόγος για τον οποίο ο 8088 αποτελούσε μια φτηνή εναλλακτική επιλογή, ήταν το εύρος του data bus. Ο 8088 ήταν 16μπιτος –μπορούσε να χειριστεί αριθμούς των 16 bits –, αλλά επικοινωνούσε μέσω ενός διαύλου των 8 bits. Ως εκ τούτου, οι αντίστοιχες μητρικές διέθεταν μικρότερο κύκλωμα και ήταν φτηνότερες. Σωστά διαβάσατε: Τα συστήματα με τον 8088 ήταν φτηνότερα από εκείνα με τον 8086, γιατί οι μητρικές διέθεταν data bus με λιγότερες γραμμές και κατασκευάζονταν ευκολότερα.

Αυτή η ιστορία μπορεί να μοιάζει άσχετη με την υλοποίηση του δικού μας υπολογιστή, αλλά δεν είναι. Όταν σταματήσαμε να σκεφτόμαστε το ποια χαρακτηριστικά θα θέλαμε να έχει κι αρχίσαμε να σκεφτόμαστε το πώς θα κατασκευαστεί, φτάσαμε γρήγορα σε μια ενοχλητική διαπίστωση (Σ.τ.Ε. "a disturbing realization", όπως λένε και στο Mayday, https://en.wikipedia.org/wiki/Mayday_(TV_series)): Η επικοινωνία των επιμέρους συστημάτων προϋπέθετε την παρουσία κάποιου διαύλου κι αυτός, με τη σειρά του, απαιτούσε πάρα πολλές κολλήσεις. Για παράδειγμα, ένας δίαυλος δεδομένων των 8bit 8 bits προϋποθέτει οκτώ γραμμές, που θα ταξιδεύουν από το

ένα υποσύστημα στο άλλο. Αν πάνω σε αυτό το δίαυλο τοποθετούσαμε και τη μνήμη RAM του υπολογιστή, θα χρειαζόμασταν αυτομάτως έναν πρόσθετο δίαυλο για τη μεταφορά των διευθύνσεων. Για παράδειγμα, για την πλήρη προσπέλαση μιας μνήμης των 32kB απαιτούνται διευθύνσεις των 15 bits κι αυτό μεταφράζεται σ' έναν δίαυλο διευθύνσεων με 15 γραμμές. Όπως αντιλαμβάνεστε, η προοπτική του να κάνουμε μερικές εκατοντάδες κολλήσεις δεν μας ενθουσίαζε καθόλου. Ακριβώς γι' αυτό, σε μια προσπάθεια να περιορίσουμε τη χαμαλοδουλειά πήραμε κάποιες αποφάσεις που μείωναν δραστικά τους διαύλους και τις κολλήσεις.

Μικροελεγκτές και επεξεργαστές

Κάθε οικογένεια μικροελεγκτών στηρίζεται σε διαφορετική αρχιτεκτονική και το ίδιο ισχύει για τους επεξεργαστές. Μη νομίζετε όμως ότι αυτή η πλούσια διαφοροποίηση "θολώνει" τη διαχωριστική γραμμή ανάμεσα στις δύο ομάδες ολοκληρωμένων. Οι μικροελεγκτές διαθέτουν όλη την απαραίτητη υποδομή για να εκτελούν κώδικα, όπως ακριβώς και οι επεξεργαστές, αλλά δεν περιορίζονται σ' αυτήν. Οι μικροελεγκτές ενσωματώνουν μόνιμο αποθηκευτικό χώρο για το εκάστοτε πρόγραμμα (flash), προσωρινή μνήμη για τα δεδομένα που θα χειρίζεται το πρόγραμμα (SRAM), καθώς και διάφορα άλλα περιφερειακά, όπως τα κυκλώματα σειριακής επικοινωνίας (UART και SPI), ο μετατροπέας ADC κ.ά. Ως εκ τούτου, μέσα σε κάθε μικροελεγκτή υπάρχει ένα data bus για τη μεταφορά δεδομένων μεταξύ των υποσυστημάτων και τουλάχιστον δύο address buses, για τον προσδιορισμό των διευθύνσεων σε κάθε τύπο μνήμης. Οι επεξεργαστές δεν ενσωματώνουν τίποτ' απ' όλα αυτά. Η προσωρινή μνήμη (RAM), ένα μόνιμο αποθηκευτικό μέσο για το BIOS (η flash του BIOS), τα υπόλοιπα αποθηκευτικά μέσα καθώς και τα περιφερειακά, αποτελούν ξεχωριστά υποσυστήματα και βρίσκονται περιμετρικά του επεξεργαστή.

Σχεδιαστικές επιλογές

Πριν ακόμα σχεδιάσουμε οποιοδήποτε κύκλωμα, ήμασταν σίγουροι ότι θα χρειαζόμασταν παραπάνω από έναν μικροελεγκτή. Το όλο σύστημα θα χαρακτηριζόταν από μεγάλη πολυπλοκότητα κι αυτό μας προκαλούσε ανασφάλεια. Κάπως έτσι, επηρεασμένοι κι από τη δομή των "κανονικών" υπολογιστών, αποφασίσαμε να χωρίσουμε την κατασκευή σε τρία τμήματα: Το κύκλωμα ήχου, το κύκλωμα γραφικών και ένα ακόμα, που θα διέθετε τον κεντρικό επεξεργαστή και κάποια βασικά "περιφερειακά". Τα τρία κυκλώματα θα ήταν ανεξάρτητα, με την έννοια ότι το καθένα θα μπορούσε να επιτελέσει όλες τις λειτουργίες του χωρίς τη βοήθεια των άλλων. Παρ' όλα αυτά, πριν καταπιαστούμε με τη σχεδίαση του καθενός, έπρεπε να λύσουμε ένα γενικότερο πρόβλημα: Πώς θα πετυχαίναμε την επικοινωνία τους;

Μετά από έναν επανέλεγχο της λίστας με τις δυνατότητες που θα θέλαμε να έχει ο υπολογιστής, καταλήξαμε σε μερικές σημαντικές αποφάσεις. Κατ' αρχάς, το κύκλωμα με τον κεντρικό επεξεργαστή θα είχε το γενικό πρόσταγμα και θα καθόριζε τι πρέπει να κάνουν τα άλλα δύο και πότε. Η "κάρτα γραφικών" δεν θα μιλούσε ποτέ με την "κάρτα ήχου", ούτε το ανάποδο. Για την ακρίβεια, αυτά τα δύο κυκλώματα δεν θα

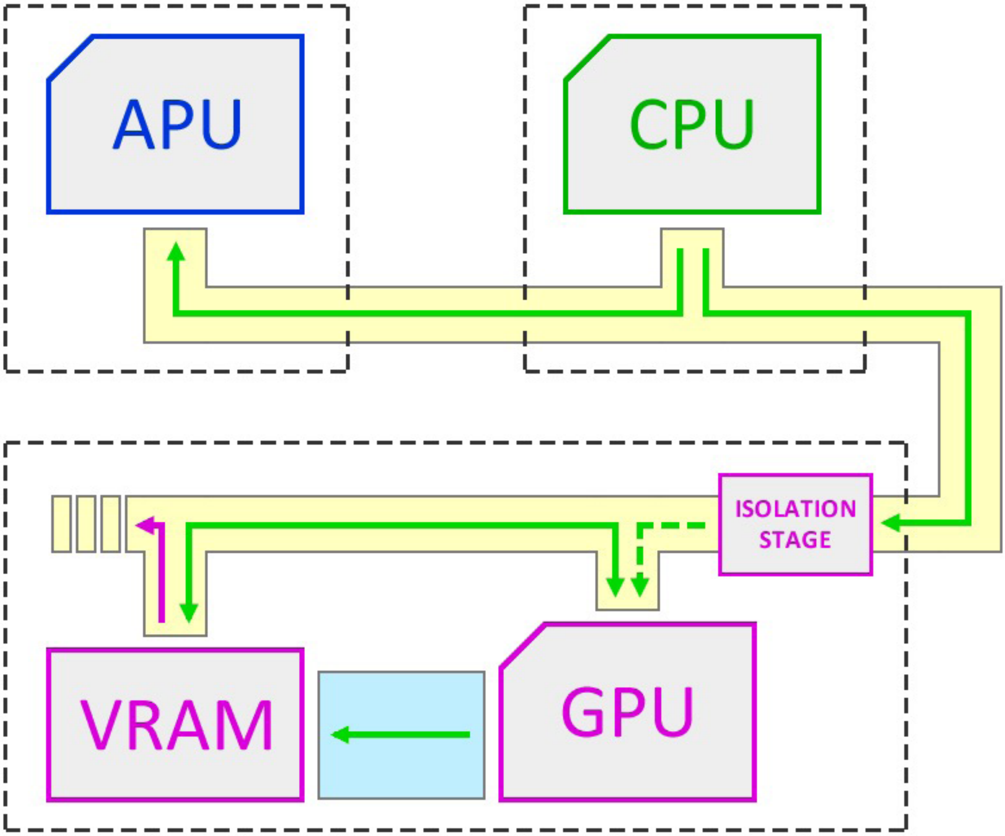
μπορούσαν να στείλουν δεδομένα σε κανέναν. Για την κεντρική μνήμη του συστήματος δεν θα χρησιμοποιούσαμε κάποιο ξεχωριστό τσιπάκι μνήμης. Θα αξιοποιούσαμε τη μνήμη SRAM του μικροελεγκτή που είχε το ρόλο του κεντρικού επεξεργαστή. Αναρωτιέστε τι σημασία έχουν όλα αυτά;

Εφόσον θα χρησιμοποιούσαμε 8μπιτους μικροελεγκτές, η επικοινωνία μεταξύ των υποσυστημάτων μπορούσε να επιτευχθεί άνετα με έναν 8μπιτο δίαυλο δεδομένων. Το γεγονός ότι η μνήμη RAM του υπολογιστή θα βρισκόταν μέσα σε έναν από τους μικροελεγκτές, μας απάλλασσε από ένα μεγάλο βραχνά: Την κατασκευή ενός πρόσθετου διαύλου για τη μεταφορά των διευθύνσεων (address bus). Επιπρόσθετα, καθώς μόνο το ένα υποσύστημα θα μπορούσε να στέλνει δεδομένα, δεν υπήρχε λόγος να κατασκευάσουμε κάποιο κύκλωμα "διαιτησίας" (bus arbitration), το οποίο θα ρύθμιζε πότε και ποιος χρησιμοποιεί τους διαύλους του συστήματος. Παρεμπιπτόντως, σε ό,τι αφορά τη χρήση των διαύλων, τα παλιά PC δεν στηρίζονταν μόνο στο κύκλωμα του bus arbitration. Παρόμοια αποστολή είχε και το κύκλωμα DMA (Direct memory Access), που επέτρεπε την πρόσβαση στη μνήμη του συστήματος χωρίς τη μεσολάβηση του κεντρικού επεξεργαστή. Η διαφορά μεταξύ των δύο έγκειται στο γεγονός ότι το DMA ρύθμιζε την πρόσβαση στους διαύλους για ένα μικρό σύνολο επίλεκτων υποσυστημάτων, ενώ το bus arbitration ήταν υπεύθυνο για τους πάντες. Ακόμα και το DMA, πριν επιτρέψει σε κάποιο υποσύστημα να χρησιμοποιήσει τους διαύλους, έπρεπε να ζητήσει (και να πάρει!) την άδεια του κυκλώματος bus arbitration. Τα συγκεκριμένα κυκλώματα εφαρμόζουν ένα σύνθετο πλέγμα κανόνων και είναι εύκολο να προκαλέσουν δυσλειτουργίες σε ολόκληρο το σύστημα. Ακριβώς γι' αυτό, η δυνατότητα να αποφύγουμε την κατασκευή τους μας ανακούφισε όσο τίποτα.

Γενική διάρθρωση

Έχοντας κατασκευάσει και στο παρελθόν κύκλωμα γραφικών, γνωρίζαμε ότι θα χρειαζόμασταν ένα τσιπάκι μνήμης που θα λειτουργούσε ως video RAM (frame buffer). Μήπως έπρεπε τελικά να φτιάξουμε κι ένα address bus, που θα συνέδεε τον κεντρικό επεξεργαστή με τη συγκεκριμένη μνήμη; Η απάντηση ήταν, ευτυχώς, όχι. Τα περιεχόμενα της συγκεκριμένης μνήμης διαβάζονται πολλές φορές το δευτερόλεπτο κι αποστέλλονται στην οθόνη για το σχηματισμό της εικόνας. Η πρόσβαση σε αυτή τη μνήμη, για οποιαδήποτε άλλη εργασία, έρχεται σε δεύτερη μοίρα κι εξελίσσεται με αργό ρυθμό. Αν επιδιώκαμε την πρόσβαση του κεντρικού επεξεργαστή στη video RAM θα προκαλούσαμε πολλές κι αισθητές καθυστερήσεις στη λειτουργία ολόκληρου του συστήματος. Έτσι, αποφασίσαμε ότι μόνον ένας μικροελεγκτής θα είχε πρόσβαση στη VRAM: Εκείνος που θα έπαιζε το ρόλο της GPU. Μήπως τώρα αναρωτιέστε πώς ξεπερνιόταν αυτή η δυσκολία στους παλιούς υπολογιστές; Στα πολύ παλιά μηχανήματα συνέβαινε ό,τι και στο δικό μας: Η πρόσβαση στη μνήμη οθόνης επιτρεπόταν μόνο στο κύκλωμα γραφικών. Αυτή η κατάσταση άλλαξε όταν οι μνήμες dual ported έγιναν φτηνότερες και πιο προσιτές. Οι συγκεκριμένες μνήμες διαθέτουν δύο εισόδους/εξόδους, οι οποίες μπορούν να λειτουργούν ταυτόχρονα. Έτσι, την ίδια στιγμή που μεταφέρονται δεδομένα από κάποια περιοχή της μνήμης προς την οθόνη, ο επεξεργαστής μπορεί να διαβάζει ή να γράφει σε μια άλλη περιοχή.

Στην εικόνα 1 φαίνεται ένα γενικό διάγραμμα του συστήματος.



Εικόνα 1
Το σύστημα διαθέτει ένα data bus που συνδέει τα τρία βασικά υποσυστήματα, όπως επίσης τη GPU με τη VRAM. Αυτός ο πολλαπλός ρόλος επέβαλε το διαχωρισμό του σε δύο ανεξάρτητα τμήματα.

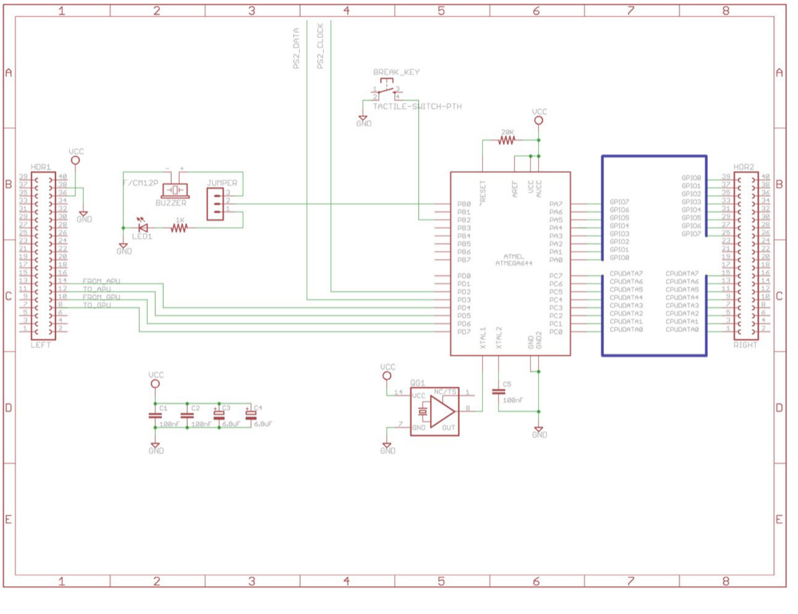
Καθένα από τα κουτιά με τις ονομασίες CPU, APU και GPU αντιστοιχεί σε έναν μικροελεγκτή, ενώ τα πλαίσια με τις διακεκομμένες γραμμές αντιστοιχούν στα τρία επιμέρους κυκλώματα του υπολογιστή. Η "διασωλήνωση" με το κίτρινο χρώμα αποτελεί το data bus του υπολογιστή. Παρατηρήστε ότι μια περιοχή του συγκεκριμένου διαύλου χρησιμοποιείται και για την επικοινωνία της GPU με τη VRAM. Τελικά, καθώς ένα τμήμα του data bus έχει πολλαπλές χρήσεις, υποχρεωνόμαστε να το χωρίσουμε στα δύο. Αυτός είναι και ο λόγος που στην "είσοδο" του κυκλώματος γραφικών υπάρχει μια βαθμίδα απομόνωσης. Θα πούμε περισσότερα γι' αυτή τη βαθμίδα, όταν φτάσουμε να εξετάσουμε το κύκλωμα γραφικών.

Η χαρά των Τρολς

Ο υπολογιστής μας έχει πολλές αρχιτεκτονικές διαφορές από τα παλιά PC και οι περισσότερες οφείλονται σε αυθαίρετες επιλογές που αποσκοπούσαν στην απλοποίηση του κυκλώματος. Οι όποιες ιδέες μας, όμως, στηρίχτηκαν στις πλούσιες δυνατότητες των μικροελεγκτών. Σε αυτό το σημείο θα μπορούσε να πάρει την πάσα ένα Τρολ και να ξεκινήσει μια αντιπαράθεση για την πλάκα του: Ποια κατηγορία ολοκληρωμένων είναι καλύτερη; Οι μικροελεγκτές ή οι επεξεργαστές; Για να αποκτήσει έστω και το παραμικρό νόημα αυτή η σύγκριση, πρέπει να περιοριστούμε στους παλιούς επεξεργαστές. Οι περισσότεροι θα έκαναν το κλασικό λάθος να στηριχτούν στα MHz. Θα έλεγαν ότι οι 8086, για παράδειγμα, χρονίζονταν το πολύ στα 8MHz κι επομένως είναι χειρότεροι, αφού οι AVR χρονίζονται έως και στα 20MHz. Κάποιοι πιο ψαγμένοι θα κατέληγαν στο ίδιο συμπέρασμα, μόνο και μόνο γιατί οι 8086 διέθεταν λιγότερες εντολές (117) από τους ATMEGA (131). Η αλήθεια όμως βρίσκεται πολύ μακριά από αυτούς τους αριθμούς. Σκεφτείτε το εξής: Αν αποφασίζαμε ότι ο υπολογιστής μας πρέπει να έχει 128KB RAM, θα χρησιμοποιούσαμε υποχρεωτικά κάποιο τσιπάκι μνήμης. Αυτή η διασύνδεση επιτυγχάνεται εύκολα, αλλά θα είχε δύο αρνητικές συνέπειες: Οι επιδόσεις του αντίστοιχου μικροελεγκτή θα μειώνονταν δραματικά και οι ακροδέκτες που θα περίσσευαν δεν θα αρκούσαν για τη σύνδεση στο data bus. Μήπως οι ATMEGA δεν είναι και τόσο ισχυροί τελικά; Τα Τρολς θα έκαναν μεγάλο γλέντι.

Κεντρική διεύθυνση

Στην εικόνα 2 φαίνεται το κύκλωμα της "μητρικής", δηλαδή του κυκλώματος που φιλοξενεί τον κεντρικό επεξεργαστή.



Εικόνα 2
Πάρα πολλές συνδέσεις –ειδικά όταν γίνονται με το κολλητήρι και πάνω σε μια διάτρητη πλακέτα–, αλλά τίποτα το ιδιαίτερο. Το κύκλωμα του "κεντρικού επεξεργαστή", σε αντίθεση με ό,τι θα περίμενε κανείς, ήταν το πιο απλό.

Εδώ μπορεί να σας κινήσει την περιέργεια η επιλογή του μικροελεγκτή ATMEGA644, που έχει πολλαπλάσιο κόστος από τον δημοφιλή ATMEGA328. Όπως αναφέραμε νωρίτερα, η μνήμη του μικροελεγκτή που βρίσκεται στο συγκεκριμένο κύκλωμα θα έχει το ρόλο της κεντρικής μνήμης του υπολογιστή. Επομένως, τα 4KB SRAM του ATMEGA644, έναντι των 2KB του ATMEGA328, αποτελούν ένα πολύ ισχυρό κίνητρο για την επιλογή του πρώτου. Έναν πρόσθετο λόγο αποτελεί και το αυξημένο πλήθος των ακροδεκτών εισόδου/εξόδου. Θυμηθείτε ότι ο υπολογιστής θα παρέχει οκτώ προγραμματιζόμενους ακροδέκτες GPIO. Όπως υποψιάζεστε, οι συγκεκριμένοι ακροδέκτες θα παρέχονται απευθείας από τον "κεντρικό μικροελεγκτή", εφόσον αυτός θα εκτελεί τον parser της BASIC και θα είναι άμεσα προγραμματιζόμενος από το χρήστη. Αρκετούς ακροδέκτες απαιτεί και η επικοινωνία των κυκλωμάτων. Για την ομαλή διεξαγωγή της, ο μικροελεγκτής πρέπει να διαθέσει οκτώ ακροδέκτες για το (8μπιτο) data bus και τέσσερεις επιπλέον. Ας δούμε το γιατί.

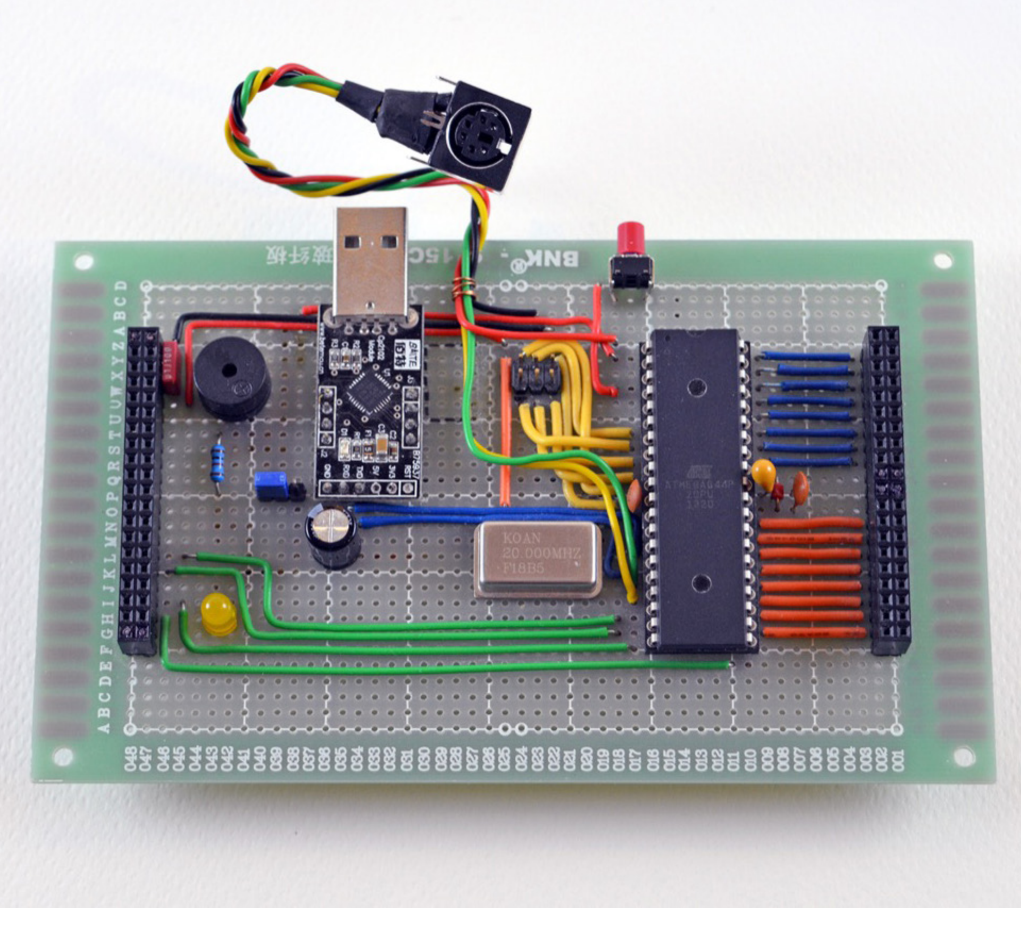
Εφόσον το σύστημα διαθέτει ένα data bus, κάθε φορά που ο επεξεργαστής "τοποθετεί" νέα δεδομένα πρέπει να ενημερώνει τον παραλήπτη με κάποιο σήμα του τύπου "Το νου σου, φίλε! Αυτά προορίζονται για 'σένα!". Κάθε παραλήπτης θα έχει τη δική του γραμμή ειδοποιήσεων και άρα χρειάζονται δύο σήματα. Στο κύκλωμα αναφέρονται ως to_GPU και to_APU. Φυσικά, το γεγονός ότι ο επεξεργαστής αποφάσισε να στείλει κάτι, δεν αποτελεί λόγο για τα περιφερειακά να διακόψουν την τρέχουσα εργασία τους. Στο κάτω κάτω, όταν η GPU παράγει το σήμα VGA ή όταν η APU παράγει κάποιο ηχητικό σήμα, μπορεί να κάνει ελάχιστα διαλείμματα και μόνο σε συγκεκριμένες χρονικές στιγμές. Πρακτικά, αυτό σημαίνει ότι ο επεξεργαστής δεν γνωρίζει αν τα δεδομένα που έστειλε ελήφθησαν και, κατ' επέκταση, δεν μπορεί να προχωρήσει στις επόμενες εργασίες του. Για την επίλυση αυτού του προβλήματος χρησιμοποιούνται δύο πρόσθετα σήματα τα οποία προέρχονται από τα περιφερειακά κυκλώματα και περιλαμβάνουν ένα μήνυμα του στιλ "Τα παρέλαβα! Ευχαριστώ πολύ!". Τα συγκεκριμένα σήματα αναφέρονται στο κύκλωμα σαν from_GPU και from_APU.

Για τα κυκλώματα

Τα κυκλώματα για τα υποσυστήματα του υπολογιστή σχεδιάστηκαν στο EAGLE (www.cadsoftusa.com/download-eagle). Πρόκειται για ένα σχεδιαστικό πρόγραμμα που παρέχεται δωρεάν και λειτουργεί χωρίς περιορισμούς –τουλάχιστον για σχετικά μικρές πλακέτες–, ενώ αποτελεί το αγαπημένο εργαλείο των απανταχού makers. Όσοι το γνωρίζουν ήδη, ενδέχεται να παρατηρήσουν ότι τα σύμβολα για τους ATMEGA644 και ATMEGA328 δεν προέρχονται από την "εργοστασιακή" βιβλιοθήκη. Αυτό συμβαίνει γιατί τα σχεδιάσαμε ξανά! Τα πρωτότυπα σύμβολα ήταν παραφορτωμένα με πολλές (και μάλλον περιττές) πληροφορίες για τους ακροδέκτες, με συνέπεια να καταντούν κουραστικά. Στη βιβλιοθήκη που δημιουργήσαμε προσθέσαμε κι ένα σύμβολο για το τσιπάκι μνήμης του κυκλώματος γραφικών. Αν θέλετε να μελετήσετε τα κυκλώματα ή να φτιάξετε την "τυπωμένη" εκδοχή τους για να κατασκευάσετε πλακέτες, μπορείτε να κατεβάσετε τα αρχεία του EAGLE από το http://bit.ly/dH047eagleschematics. Στο πακέτο θα βρείτε τα τρία κυκλώματα, καθώς και την βιβλιοθήκη μας.

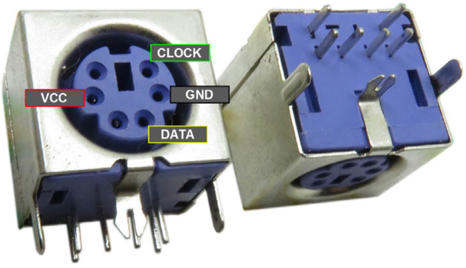
Κάθε φορά που προκύπτει οποιοδήποτε πρόβλημα (πάτημα του backspace ενώ ο δρομέας βρίσκεται ήδη στην αρχή της γραμμής, πάτημα πλήκτρου ενώ η buffer του πληκτρολογίου έχει γεμίσει κ.λπ.), όπως και κατά την εκκίνηση του συστήματος, ακούγεται ένα τυπικό μπιπ. Αν ο ήχος κριθεί ενοχλητικός, ο χρήστης μπορεί να επιλέξει την ενεργοποίηση ενός LED, αντί του buzzer. Η σχετική ρύθμιση (ενεργοποίηση του λεγόμενου "visible bell") πραγματοποιείται από ένα jumper. Μπορεί ο υπολογιστής μας να μην παίρνει βραβείο ευχρηστίας, αλλά πρέπει να του αναγνωρίσετε ότι προσπαθεί.

Στη φωτογραφία της πλακέτας με το κύκλωμα του επεξεργαστή (εικόνα 3) μπορείτε να διακρίνετε την υποδοχή PS2, ένα κόκκινο κουμπάκι πάνω από τον μικροελεγκτή, ένα σύμπλεγμα κίτρινων καλωδίων κι ένα ολόκληρο κύκλωμα "USB to serial".



Εικόνα 3
Μεταξύ άλλων, στη φωτογραφία της "μητρικής" φαίνεται κι ένα κύκλωμα USB-to-Serial. Υποτίθεται ότι η παρουσία του θα ήταν προσωρινή, αλλά ξέρετε πώς είναι αυτά. Το αξιοποιήσαμε σαν είσοδο δεδομένων, πριν προστεθεί ο κώδικας για τη σύνδεση του πληκτρολογίου, αλλά παρέμεινε.

Το διακοπτάκι προστέθηκε στη φάση της αποσφαλμάτωσης του parser της BASIC και λειτουργεί σαν σήμα διακοπής του εκτελούμενου προγράμματος. Μπορείτε να το φαντάζεστε σαν το ισοδύναμο του συνδυασμού πλήκτρων [CTLR+C]. Βέβαια το πάτημα αυτού του κουμπιού λαμβάνεται υπόψη άμεσα, χωρίς τη μεσολάβηση του κώδικα που "διαβάζει" το πληκτρολόγιο. Το κύκλωμα "USB to serial" αποτελεί ένα ακόμα απομεινάρι της διαδικασίας αποσφαλμάτωσης του firmware. Το συγκεκριμένο κύκλωμα αποτελούσε το μοναδικό αξιόπιστο μέσο εισόδου, πριν τελειοποιηθεί η επικοινωνία με το πληκτρολόγιο PS2. Σε εκείνο το στάδιο, ο μόνος τρόπος αλληλεπίδρασης με τον parser της BASIC ήταν μέσα από μια απλή σειριακή σύνδεση. Τέλος, τα κίτρινα καλώδια καταλήγουν σε ένα pin header για τον εύκολο προγραμματισμό του μικροελεγκτή (ISP από το In-System Programming). Όπως αντιλαμβάνεστε, τα συγκεκριμένα στοιχεία του κυκλώματος θα μπορούσαν και να λείπουν, χωρίς να αλλάζει κάτι σημαντικό στη λειτουργία του υπολογιστή. Το ακριβώς αντίθετο ισχύει για το μεταλλικό κουτάκι, στα αριστερά του μικροελεγκτή. Πρόκειται για έναν ταλαντωτή που παράγει τετραγωνικούς παλμούς στα 20MHz. Με άλλα λόγια, αποτελεί το "ρολόι" του επεξεργαστή.



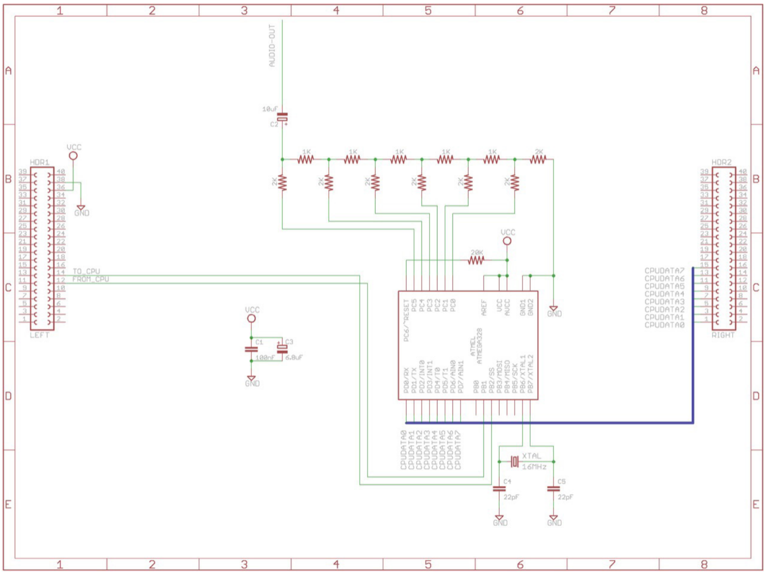
Εικόνα 4
Ο υπολογιστής μας συνδέεται με το πληκτρολόγιο μέσω μιας τυπικής υποδοχής PS2. Το γενικό όνομα αυτών των υποδοχών είναι mini-DIN 6 και στην εικόνα έχουμε επισημάνει τους ακροδέκτες που αξιοποιούνται από το πρωτόκολλο PS2. Το "λογικό" μέρος της σύζευξης θα μας απασχολήσει όταν καταπιαστούμε με το software.

Μα, τι κάνει επιτέλους αυτό το "ρολόι";

Φανταστείτε ένα σύνθετο ψηφιακό κύκλωμα με δεκάδες πύλες, που η έξοδος της μίας μπορεί να τροφοδοτεί άλλες δύο-τρεις, ενώ κάποιες από αυτές ενδέχεται να επιστρέφουν το σήμα της εξόδου τους σε κάποια προηγούμενη πύλη και πάει λέγοντας. Σε ένα τέτοιο σύστημα προκύπτει το εξής πρόβλημα: Πότε μπορούμε να θεωρήσουμε ότι τα σήματα που παράγονται είναι έγκυρα; Οι πύλες, βλέπετε, δεν λειτουργούν ακαριαία. Αυτό σημαίνει ότι τα σήματα που φτάνουν σε μια βαθμίδα του κυκλώματος ενδέχεται να μην έχουν οριστικοποιηθεί ακόμα, καθώς οι προηγούμενες πύλες δεν έχουν ολοκληρώσει την εργασία τους. Αυτή η ασάφεια αποφεύγεται με τη χρήση ενός ειδικού παλμού, ο οποίος ενημερώνει τις πύλες για το πότε θεωρούνται έγκυρα τα διάφορα σήματα. Ο εν λόγω παλμός ονομάζεται παλμός συγχρονισμού (synchronization pulse ή clock) και το ειδικό κύκλωμα που τον παράγει είναι το περίφημο ρολόι (clock source). Το πόσο μπορεί να αυξηθεί η συχνότητα αυτού του παλμού καθορίζεται από την ταχύτητα των επιμέρους στοιχείων που απαρτίζουν ένα κύκλωμα και, σε τελική ανάλυση, από την ταχύτητα λειτουργίας των λογικών πυλών. Με τη σειρά της, αυτή η ταχύτητα καθορίζεται από το χρόνο που απαιτεί ένα σήμα για να ολοκληρώσει τη διαδρομή του εντός της πύλης (propagation delay time).

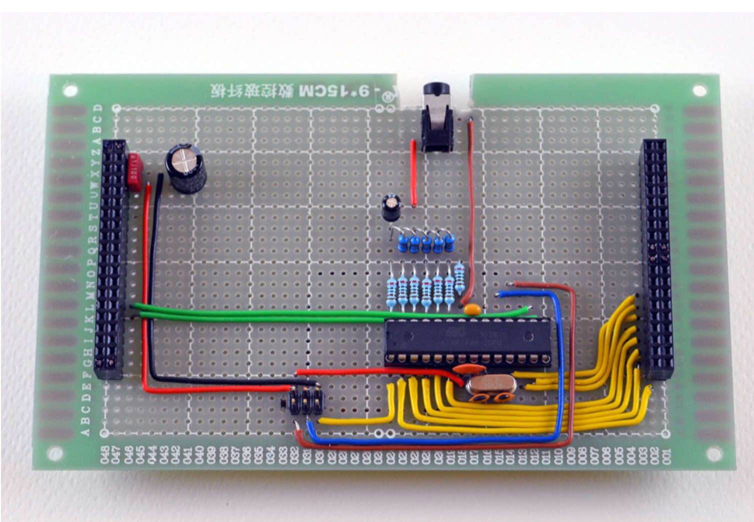
Παραγωγή ήχου

Όπως μπορείτε να διαπιστώσετε από την εικόνα 5, το κύκλωμα του ήχου είναι εξίσου μικρό με εκείνο της μητρικής.



Εικόνα 5
Το κύκλωμα της "κάρτας ήχου" είναι το μικρότερο απ' όλα, αλλά παρουσιάζει αρκετό ενδιαφέρον. Πάνω από τον επεξεργαστή δεσπόζει ένα δικτύωμα αντιστάσεων R/2R, που συμπεριφέρεται σαν DAC των 6 bits.

Αυτή τη φορά, εφόσον οι απαιτήσεις σε ακροδέκτες ήταν ελάχιστες, χρησιμοποιήσαμε τον ATMEGA328. Ο μεγάλος ATMEGA644 θα αύξανε το κόστος και δεν θα προσέφερε τίποτα. Στη φωτογραφία του κυκλώματος (εικόνα 6) ξεχωρίζει αμέσως η απόληξη του data bus του συστήματος, που έχει υλοποιηθεί με τα κίτρινα καλώδια.



Εικόνα 6
Τα κίτρινα καλώδια αποτελούν τμήμα του data bus, που συνδέει τη CPU με την APU, το τσιπάκι του ήχου! Τα δύο πράσινα καλώδια χρησιμοποιούνται επικουρικά, για την ομαλή μεταφορά δεδομένων μέσω του διαύλου.

Οι δύο συνδέσεις που ξεκινούν από τον μικροελεγκτή και καταλήγουν στο αριστερό Pin header, μεταφέρουν τα σήματα ελέγχου που ρυθμίζουν την επικοινωνία με τον κεντρικό επεξεργαστή. Στο κύκλωμα της μητρικής τα εν λόγω σήματα ονομάζονταν TO_APU και FROM_APU, ενώ σε αυτό το κύκλωμα έχουν βαφτιστεί FROM_CPU και TO_CPU αντίστοιχα. Αυτή η μεταβολή στην ονομασία των σημάτων μοιάζει παραπλανητική, αλλά στην πράξη αποδείχθηκε χρήσιμη. Όταν γράφαμε τον κώδικα και ρυθμίζαμε τους ακροδέκτες του μικροελεγκτή, ήταν προτιμότερο να βλέπουμε τη φορά των σημάτων σε σχέση με τον ίδιο και όχι σε σχέση με τον κεντρικό επεξεργαστή. Όλα αυτά όμως έχουν πολύ λιγότερο ενδιαφέρον από το κύκλωμα των αντιστάσεων που βρίσκεται πάνω από τον μικροελεγκτή. Η συγκεκριμένη διάταξη ονομάζεται R/2R και αποτελεί έναν μετατροπέα ψηφιακών σημάτων σε αναλογικά (DAC). Η δική μας υλοποίηση μπορεί να μετατρέπει ψηφιακά σήματα των 6bit. Αναρωτιέστε από πού προέκυψε αυτό το έξι; Η απάντηση προϋποθέτει μια μικρή ανάλυση.

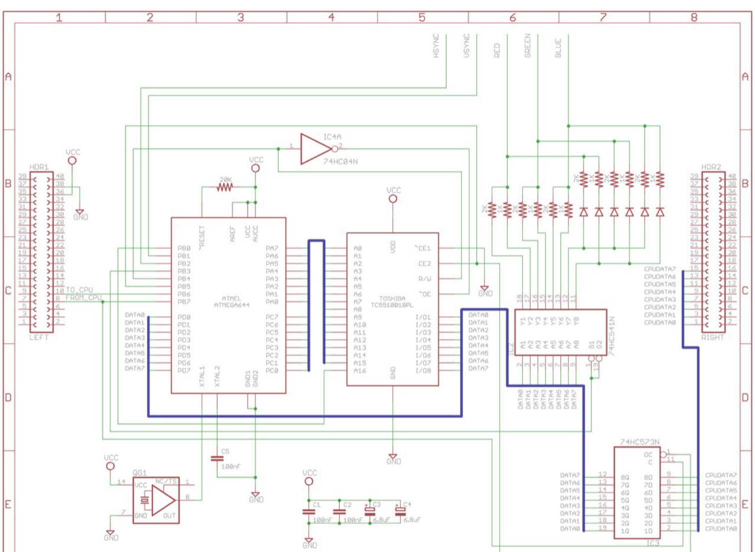
Στο εισαγωγικό άρθρο της σειράς μας, που αρχίζει από τη σελίδα 14 του τεύχους, αναφέραμε ότι ο υπολογιστής μας διαθέτει τέσσερα κανάλια ήχου. Η συγκεκριμένη επιλογή ήταν αυθαίρετη αλλά όχι και τυχαία. Μερικοί διάσημοι υπολογιστές της δεκαετίας του 80 είναι ο Atari ST, ο οποίος διέθετε τρία κανάλια ήχου, η Amiga 500 με τέσσερα κανάλια κι ο Commodore 64 με επίσης τέσσερα κανάλια ήχου. Όπως υποψιάζεστε, αυτό που επιδιώκαμε ήταν να μιμηθούμε τα θρυλικά μηχανήματα του παρελθόντος. Βέβαια κάθε κανάλι συνεπάγεται την ύπαρξη ενός synthesizer και, εν προκειμένω, θα χρειαζόμασταν τέσσερα synthesizers. Τα ψηφιακά synthesizers δημιουργούν ήχους υπολογίζοντας τα κατάλληλα δείγματα (αριθμητικές τιμές), που όταν "μπουν στη σειρά" σχηματίζουν τη ζητούμενη κυματομορφή. Ακριβώς γι' αυτό, ένα θεμελιώδες χαρακτηριστικό των ψηφιακών synthesizers αποτελεί το πλήθος των bits που χρησιμοποιούν για την περιγραφή των δειγμάτων. Ουσιαστικά, το πλήθιος των bits καθορίζει την ακρίβεια με την οποία μπορεί το synthesizer να περιγράφει τις διαδοχικές στάθμες σε μια κυματομορφή. Για τα δικά μας synthesizers, που θα υλοποιούνται στο software και δεν υπάρχουν πολλά περιθώρια, επιλέξαμε το εύρος των 4 bits. Κρατήστε τα όλα αυτά κι ας πάμε λίγο παραπέρα. Η ψηφιακή μίξη του ήχου μερικών synthesizers επιτυγχάνεται με την αριθμητική πρόσθεση των δειγμάτων που παράγονται κάθε στιγμή. Για να ακούσουμε το τελικό σήμα του μίκτη πρέπει να δώσουμε τις διαδοχικές τιμές που υπολογίζει σε κάποιον DAC. Στη συνέχεια, οι αντίστοιχες στάθμες τάσης που δημιουργεί ο DAC οδηγούνται σε κάποιον ενισχυτή και τελικά στα ηχεία. Προσέξτε τώρα το εξής: Η μέγιστη τιμή που μπορεί να πάρει η έξοδος του μίκτη καθορίζει το πλήθος των bits που πρέπει να είναι σε θέση να δεχτεί ο DAC. Στον δικό μας υπολογιστή, κάθε κανάλι είναι τετράμπιτο και άρα η μέγιστη τιμή για τα δείγματα είναι ο αριθμός 15. Συνεπώς, εφόσον έχουμε τέσσερα συνθεσάιζερ (ένα για κάθε κανάλι), η μέγιστη τιμή στην έξοδο του μίκτη είναι ο αριθμός 60. Τώρα θα σας ρωτούσαμε πόσα bits απαιτούνται για την αναπαράσταση αυτού του αριθμού στο δυαδικό, αλλά σας έχουμε δώσει την απάντηση ήδη. Χρειάζονται έξι – όσα είναι και τα bits που δέχεται στην είσοδό του ο DAC ;)

Τι εστί μετατροπή;

Η συζήτηση γύρω από το πλήθος των bits (δηλαδή γύρω από το εύρος) του DAC, είναι πιθανό να δημιούργησε μια απορία: Γιατί μας ενδιαφέρει το πλήθος των bits που δέχεται στην είσοδό του; Σε τελική ανάλυση, τι ακριβώς πετυχαίνει ένας DAC; Αυτά τα ερωτήματα μοιάζουν αρκετά απλά, αλλά μη νομίζετε ότι γνωρίζουν όλοι τις απαντήσεις. Ας πάρουμε τα πράγματα με τη σειρά. Οι DAC δέχονται στην είσοδό τους αριθμούς, εκφρασμένους στο δυαδικό σύστημα. Αυτοί οι αριθμοί ανήκουν στο κλειστό διάστημα από μηδέν έως 2n - 1, όπου το n είναι το πλήθος των bits που δέχεται ο εκάστοτε DAC. Για κάθε αριθμό σε αυτό το διάστημα ο DAC παράγει μια ανάλογη τάση στο διάστημα από 0 έως K volt, όπου το K αποτελεί την τάση τροφοδοσίας του κυκλώματος. Στην περίπτωση του κυκλώματος ήχου του δικού μας υπολογιστή, ο DAC δέχεται αριθμούς των 6bits (από μηδέν έως 63) και παράγει μια ανάλογη τάση στο διάστημα από 0 έως 5Volt. Για παράδειγμα, αν τον τροφοδοτήσουμε με τον αριθμό 31 θα δημιουργήσει μια τάση 2.5Volt, αν δεχτεί τον αριθμό 26 θα δημιουργήσει μια τάση κοντά στα 2Volt, ενώ αν του δώσουμε τον αριθμό 1 θα δημιουργήσει μια τάση κοντά στα 0.078V, το ένα εξηκοστό τέταρτο των 5V.

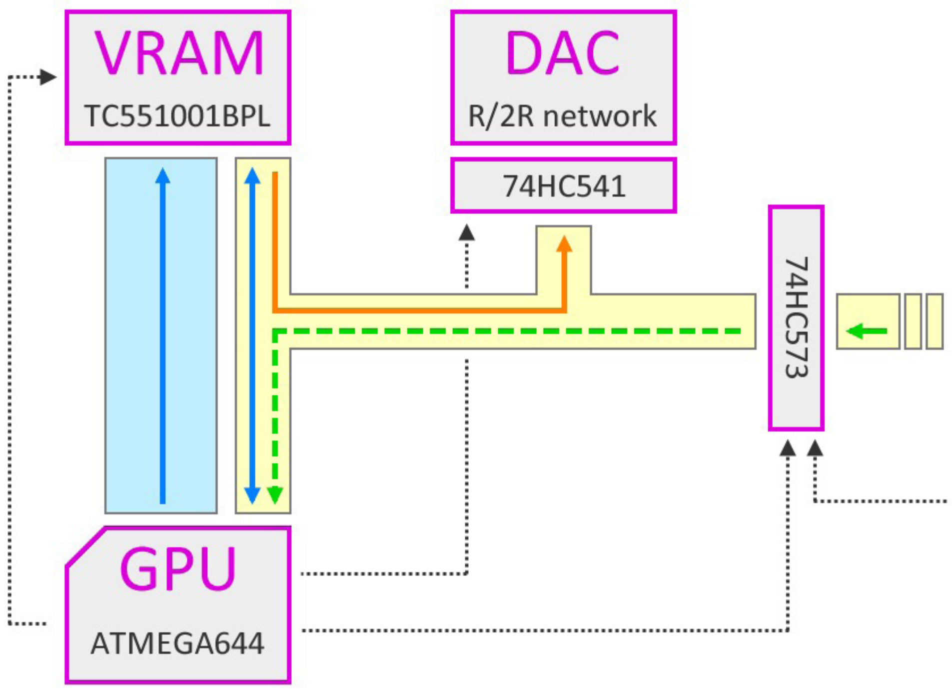
Παραγωγή εικόνας

Το κύκλωμα γραφικών φαίνεται στην εικόνα 7 κι εκ πρώτης όψης φαντάζει αρκετά περίπλοκο. Στην πραγματικότητα η περισσότερη δουλειά πραγματοποιείται πάλι στο software και το κύκλωμα ακολουθεί μια πολύ απλή λογική.



Εικόνα 7
Το κύκλωμα της "κάρτας γραφικών" είναι μεγάλο κι εξίσου ενδιαφέρον. Ενσωματώνει το μηχανισμό (74HC573) που χωρίζει το data bus του συστήματος στα δύο, τη μνήμη οθόνης (TC551001BPL) και έναν DAC, που μετατρέπει τα δεδομένα των pixels σε τρεις αναλογικές τάσεις – μία για κάθε χρωματική συνιστώσα.

Αυτό που ενδέχεται να σας τρομάζει είναι το πλήθος των συνδέσεων. Πιστεύουμε ότι το διάγραμμα της εικόνας 8 ξεκαθαρίζει το τοπίο σε μεγάλο βαθμό.

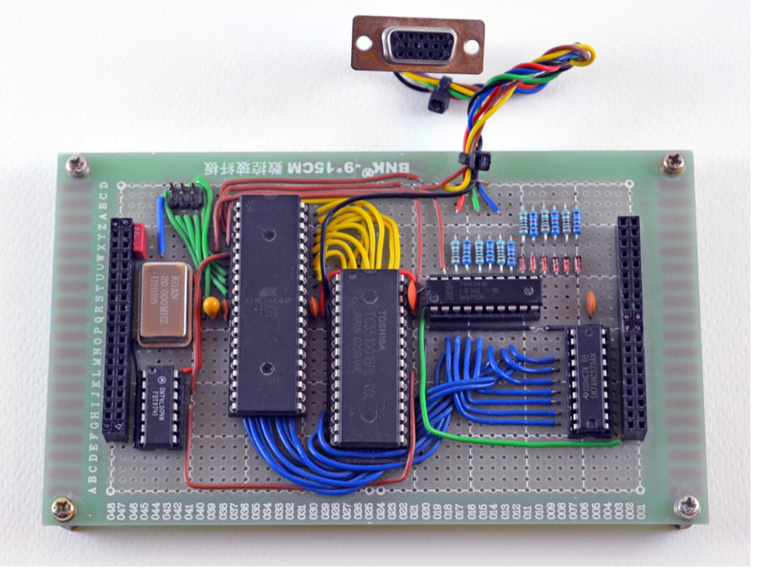


Εικόνα 8
Η σύνδεση της GPU με τη μνήμη οθόνης προϋποθέτει δύο διαύλους (data και address). Ο ένας από αυτούς (data) χρησιμοποιείται και για την επικοινωνία με τον επεξεργαστή, όπως επίσης και για την έξοδο των δεδομένων προς την οθόνη. Οι διακεκομμένες γραμμές αναπαριστούν τα σήματα ελέγχου που ανοιγοκλείνουν τις δύο "στρόφιγγες".

Όπως είπαμε νωρίτερα, το κύκλωμα γραφικών περιλαμβάνει ένα τσιπάκι μνήμης που έχει το ρόλο της μνήμης οθόνης. Στο συγκεκριμένο κύκλωμα, λοιπόν, το data bus αποκτά διπλή αποστολή: Συνδέει τον μικροελεγκτή του κυκλώματος με τον κεντρικό επεξεργαστή κι επιπρόσθετα με τη video RAM. Για την πρώτη σύνδεση προβλέπεται μόνο η μία φορά κίνησης των δεδομένων (από τον επεξεργαστή προς τη GPU), ενώ για τη δεύτερη σύνδεση προβλέπεται η αμφίδρομη μεταφορά δεδομένων (ο μικροελεγκτής μπορεί να διαβάζει αλλά και να γράφει δεδομένα στη μνήμη). Η πολλαπλή χρήση του διαύλου διευκολύνει την κατασκευή του κυκλώματος –φανταστείτε τι θα γινόταν να είχαμε δύο data buses–, αλλά εισάγει ένα πρόβλημα: Τι θα συμβεί στην περίπτωση που ο επεξεργαστής τοποθετήσει δεδομένα στο data bus, την ίδια στιγμή που η GPU προσπαθεί να διαβάσει ή να γράψει κάτι στη μνήμη; Αυτό το ζήτημα αντιμετωπίζεται εύκολα με το "σπάσιμο" του διαύλου σε δύο ανεξάρτητα τμήματα. Για το σκοπό αυτό επιστρατεύσαμε το ολοκληρωμένο 75HC573. Το εν λόγω τσιπάκι ενσωματώνει έναν προσωρινό αποθηκευτικό χώρο (latch) κα-

θώς και μια βαθμίδα εξόδου, η οποία μπορεί να θέτει τους ακροδέκτες της σε τρεις καταστάσεις: Low, High και Hi-Z. Η τρίτη κατάσταση ισοδυναμεί με την αποσύνδεση του αντίστοιχου ακροδέκτη από το κύκλωμα. Ένας ακροδέκτης που έχει τεθεί σε αυτή την κατάσταση δεν επηρεάζει την κατάσταση των ακροδεκτών με τους οποίους συνδέεται. Προσέξτε τώρα το εξής: Όταν ο επεξεργαστής θέλει να στείλει δεδομένα στο κύκλωμα γραφικών, τα τοποθετεί στο data bus και στέλνει μια σχετική ειδοποίηση στη GPU. Η ίδια ειδοποίηση (το ίδιο σήμα) αξιοποιείται και για την ενεργοποίηση του latch του 74HC573. Με αυτόν τον τρόπο τα δεδομένα του επεξεργαστή αποθηκεύονται πλέον στο εσωτερικό του 74HC573 κι ο δίαυλος μπορεί να αξιοποιηθεί για άλλες δουλειές, π.χ., για την αποστολή δεδομένων στο κύκλωμα ήχου. Όταν η GPU αποφασίσει ότι μπορεί να παραλάβει τα δεδομένα του επεξεργαστή, αρκεί να ενεργοποιήσει τις εξόδους του 74HC573 και, φυσικά, να ειδοποιήσει τον επεξεργαστή ότι παρέλαβε το πακετάκι. Με αυτή τη διαδικασία ο επεξεργαστής μπορεί να στέλνει οτιδήποτε στο κύκλωμα γραφικών, χωρίς να παρεμβαίνει στην επικοινωνία μεταξύ GPU και video RAM.

Το ολοκληρωμένο 74HC541 αποτελεί μια απλουστευμένη εκδοχή του 74HC573. Για την ακρίβεια ενσωματώνει μια βαθμίδα εξόδου τριών καταστάσεων, αλλά στερείται του εσωτερικού αποθηκευτικού χώρου. Η ύπαρξή του επιβάλλεται από τις προδιαγραφές του σήματος VGA. Σύμφωνα με αυτές, όταν δεν εκπέμπεται πληροφορία εικόνας (αυτό συμβαίνει στην αρχή και στο τέλος κάθε γραμμής σάρωσης, όπως επίσης στην αρχή και στο τέλος κάθε frame), τότε τα τρία σήματα για το χρώμα πρέπει να μηδενίζονται. Στο δικό μας κύκλωμα αυτά τα σήματα μεταφέρονται από το data bus, το οποίο γενικά μεταφέρει διάφορα άλλα δεδομένα. Εδώ έρχεται να μας βοηθήσει το 74HC541, που λειτουργεί σαν στρόφιγγα. Όταν οι έξοδοί του ενεργοποιούνται –κάτι που αποφασίζει η GPU–, τα δεδομένα του data bus προχωρούν προς το DAC κι από 'κεί προς την οθόνη. Διαφορετικά, ανεξάρτητα από το τι γίνεται στο data bus, η οθόνη δεν λαμβάνει τίποτα.



Εικόνα 9
Σε αυτό το κύκλωμα χρησιμοποιήσαμε μπλε καλώδια για το data bus και κίτρινα για το address bus. Πριν σκεφτείτε ότι η κατασκευή αυτού του κυκλώματος απαιτεί πολύ κόπο, σημειώστε ότι ένα τμήμα του address bus βρίσκεται στην άλλη όψη της πλακέτας.

Μα, τι γίνεται εκεί μέσα;

Είμαστε σίγουροι ότι η παρουσίαση του κυκλώματος ήχου έχει αφήσει αναπάντητο ένα μεγάλο ερώτημα: Πώς καταφέρνει ένας σωρός αντιστάσεων να μετατρέπει ένα αριθμητικό μέγεθος σε μια τάση; Στην εικόνα 10 φαίνεται το κύκλωμα ενός τετράμπιτου DAC. Ο συγκεκριμένος μετατροπέας είναι σχεδιασμένος λίγο διαφορετικά, αλλά πέρα από το μειωμένο πλήθος των εισόδων (4 αντί για 6) δεν έχει καμία διαφορά από τον μετατροπέα του κυκλώματος ήχου. Για να κατανοήσουμε τη λειτουργία του πρέπει να δούμε τη σχέση που συνδέει την τάση εξόδου με την κατάσταση των εισόδων. Καθεμία από τις εισόδους μπορεί να πάρει δύο διακριτές τιμές (0 και 1). Με άλλα λόγια, κάθε είσοδος μπορεί να αντιμετωπισθεί σαν μια πηγή τάσης, που παρέχει 0 ή 5V (ή όποια άλλη τάση αντιστοιχεί στη λογική κατάσταση 1). Έχοντας αυτό κατά νου, η έξοδος του μετατροπέα προκύπτει από την ακόλουθη σχέση:

$$V_{out} = V_{bit0}/16 + V_{bit1}/8 + V_{bit2}/4 + V_{bit3}/2$$

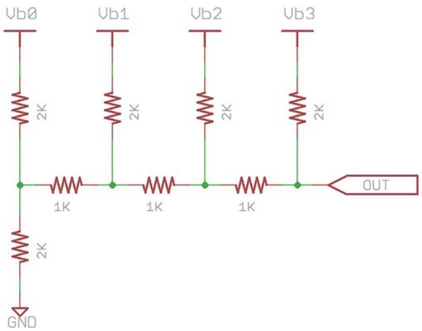
Η σχέση δείχνει ξεκάθαρα ότι κάθε είσοδος του τετράμπιτου DAC επιδρά στην έξοδο με τόση "ένταση", όση είναι και η αξία του αντίστοιχου bit σε σχέση με τα υπόλοιπα. Τώρα όμως μπορεί να γεννήθηκε ένα νέο ερώτημα: Πώς προκύπτει αυτή η –ομολογουμένως βολική– σχέση; Ο υπολογισμός της εξόδου του κυκλώματος πραγματοποιείται εύκολα, με τη βοήθεια των θεωρημάτων της υπέρθεσης (ή επαλληλίας) και του Thevenin. Σύμφωνα με το θεώρημα της υπέρθεσης, σε ένα κύκλωμα σαν αυτό που εξετάζουμε (γραμμικό), η τάση εξόδου που παράγεται από την επίδραση πολλών διαφορετικών πηγών τάσης (τα επιμέρους bits), προκύπτει από το άθροισμα των επιμέρους τάσεων εξόδου που θα παρήγαγε κάθε πηγή τάσης αν επιδρούσε μόνη της (αν οι υπόλοιπες ήταν βραχυκυκλωμένες). Η παραπάνω παρουσίαση του θεωρήματος είναι πολύ απλουστευμένη, αλλά είναι επαρκής για τις ανάγκες του άρθρου. Στην εικόνα 11 βλέπουμε τα βήματα για τον υπολογισμό της τάσης εξόδου που δημιουργεί η πηγή Vb1, όταν όλες οι υπόλοιπες είναι βραχυκυκλωμένες. Στην πορεία της ανάλυσης εφαρμόζουμε το θεώρημα Thevenin τρεις φορές, για την περαιτέρω απλοποίηση του κυκλώματος. Εν τέλει, όπως φαίνεται και στο τελευταίο καρέ της εικόνας, καταλήγουμε στην ακόλουθη σχέση:

$$V_{out1} = V_{b1}/8$$

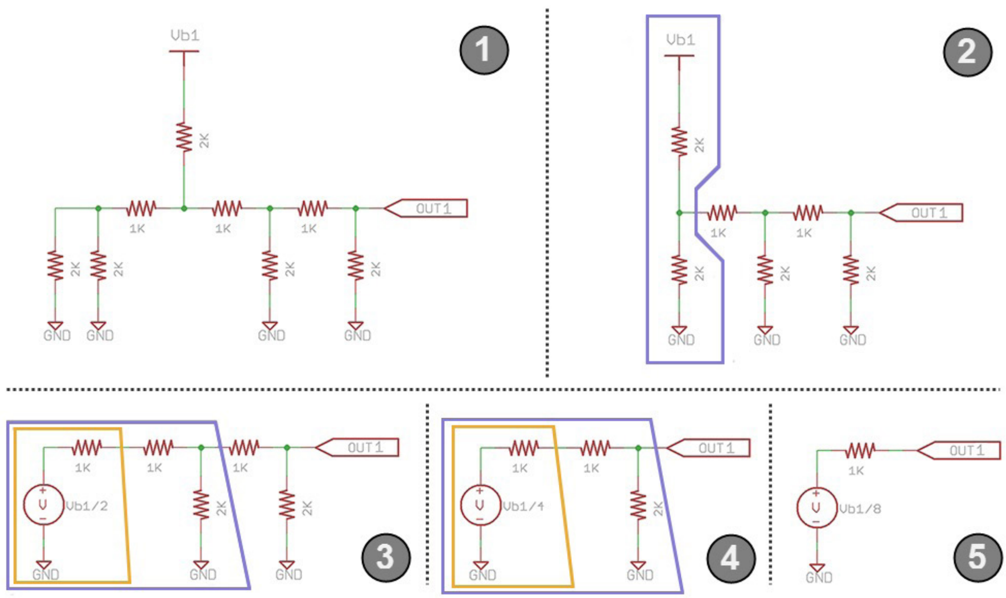
Με παρόμοιο τρόπο μπορούμε να υπολογίσουμε και την τάση εξόδου που δημιουργεί καθεμία από τις υπόλοιπες πηγές (εισόδους). Στη συνέχεια, αρκεί να αντικαταστήσουμε τα ευρήματά μας στη μαθηματική έκφραση του θεωρήματος της υπέρθεσης:

$$V_{out} = V_{out0} + V_{out1} + V_{out2} + V_{out3}$$

Αν γίνουν σωστά οι υπολογισμοί, θα φτάσουμε στη σχέση που χρησιμοποιήσαμε αρχικά για την περιγραφή της συμπεριφοράς του DAC.



Εικόνα 10
Ένα δικτύωμα R/2R, παρόμοιο με εκείνο που χρησιμοποιείται στο κύκλωμα ήχου. Η μόνη τους διαφορά έγκειται στο πλήθος των εισόδων και στον τρόπο που έχουν σχεδιαστεί. Θέλουμε να πιστεύουμε ότι αυτό "διαβάζεται" ευκολότερα.



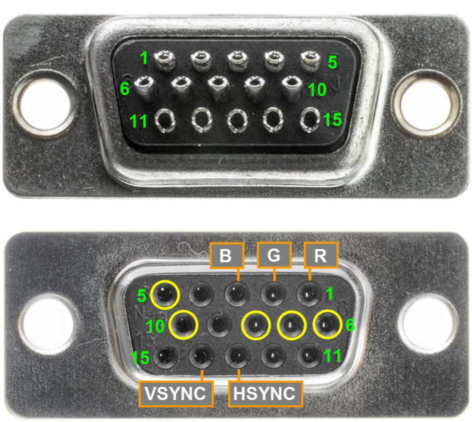
Εικόνα 11
Το θεώρημα της υπέρθεσης (superposition) επιτρέπει να υπολογίσουμε την έξοδο του κυκλώματος, αθροίζοντας την επίδραση που έχει κάθε είσοδος ξεχωριστά. Με τη μπλε γραμμή έχουμε μαρκάρει τα τμήματα στα οποία εφαρμόσαμε το θεώρημα Thevenin. Τα πορτοκαλί πλαίσια περιλαμβάνουν τα αντίστοιχα "ισοδύναμα Thevenin", δηλαδή τα προϊόντα από την εφαρμογή του ομώνυμου θεωρήματος.

Αληθινό χρώμα

Η σκέψη ότι όσα βλέπουμε στην οθόνη υπάρχουν αποθηκευμένα και στη video RAM, δεν μας προκαλεί καμία αίσθηση. Το κόστος της μνήμης είναι τόσο χαμηλό πλέον, που ακόμα και τα πιο "μικρά" κυκλώματα γραφικών έχουν πρόσβαση σε εκατοντάδες MB μνήμης. Σήμερα, ακόμα και στην περίπτωση των μεγάλων αναλύσεων, υπάρχει άφθονος χώρος για την αποθήκευση του χρώματος κάθε pixel. Αρκεί να σκεφτεί κανείς ότι για εικόνα ανάλυσης 1920x1200 και με 24bit χρώμα, αρκούν λιγότερα από 7MB. Ωστόσο, μερικές δεκαετίες πριν τα πράγματα ήταν εντελώς δια-

φορετικά. Τα τσιπάκια μνήμης προσέφεραν πολύ μικρότερο χώρο και σε εξαιρετικά υψηλότερο κόστος, ενώ και τα κυκλώματα γραφικών ήταν τόσο αργόστροφα που δεν θα μπορούσαν να χειριστούν μεγαλύτερους όγκους δεδομένων. Για να πάρετε μια ιδέα, τα κυκλώματα γραφικών CGA και EGA χρησιμοποιούσαν 4bit για την περιγραφή κάθε pixel. Αυτό σημαίνει ότι κάθε στιγμή μπορούσαν να προβάλλουν εικόνες των 16 χρωμάτων μόλις. Ε, λοιπόν, ο δικός μας υπολογιστής χρησιμοποιεί 8 bits για κάθε pixel κι αυτό σημαίνει ότι μπορεί να προβάλλει 256 διαφορετικές αποχρώσεις. Επιπρόσθετα, το δικό μας κύκλωμα γραφικών είναι true color! Τι, δεν μας πιστεύετε;

Σε ορισμένα παλιά συστήματα, τα δεδομένα που αποθηκεύονταν στη μνήμη οθόνης δεν περιέγραφαν το χρώμα του αντίστοιχου pixel. Τα αποθηκευμένα bits λειτουργούσαν ως δείκτης προς την περιγραφή κάποιου χρώματος, που βρισκόταν σε μια άσχετη περιοχή της μνήμης. Αυτή η περιοχή της μνήμης ονομαζόταν παλέτα, ενώ η τεχνική ονομασία της ήταν Color Look-Up Table (CLUT). Σε αυτό το σύστημα στηρίζονταν και τα κυκλώματα EGA: Διέθεταν μια ειδική περιοχή της μνήμης, στην οποία βρίσκονταν αποθηκευμένες 64 διαφορετικές αποχρώσεις. Αυτά τα χρώματα ήταν χωρισμένα σε τέσσερεις ομάδες των δεκαέξι και το κύκλωμα γραφικών μπορούσε να χρησιμοποιήσει μόνο μια ομάδα για κάθε καρέ (frame). Τελικά, αν και η παλέτα ήταν εξάμπιτη (26 = 64), η μνήμη οθόνης περιλάμβανε μόνο 4 bits για κάθε pixel. Αυτή η τετράδα λειτουργούσε σαν δείκτης προς μια από τις δεκαέξι αποχρώσεις της επιλεγμένης ομάδας. Έτσι, η οθόνη μπορούσε να προβάλλει μόνο 16 χρώματα σε κάθε frame, αν και η παλέτα του συστήματος διέθετε συνολικά 64. Στο δικό μας κύκλωμα γραφικών, όπως και στα περισσότερα, τα bits που αποθηκεύονται για κάθε pixel είναι μοιρασμένα στις τρεις χρωματικές συνιστώσες (Red, Green, Blue). Επομένως, τα δεδομένα της μνήμης οθόνης προσδιορίζουν "απευθείας" τις διάφορες αποχρώσεις, χωρίς την ανάγκη κάποιας παλέτας. Στο παρελθόν, αυτό το σύστημα ονομαζόταν true color.



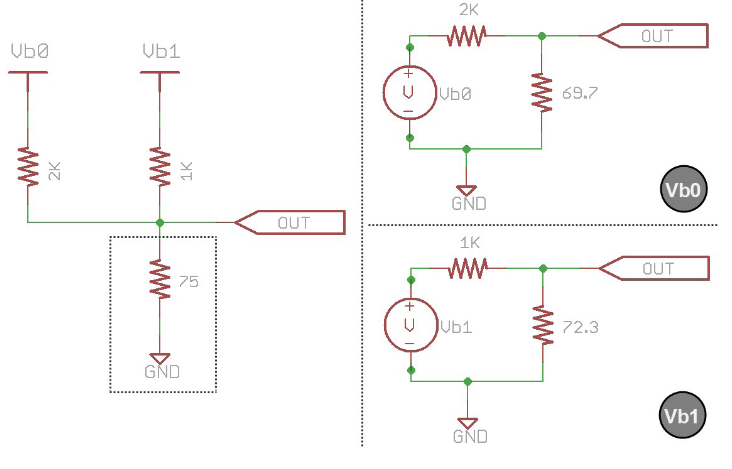
Εικόνα 12.
Μια σύνδεση VGA απαρτίζεται από πέντε κρίσιμα σήματα. Τα πρώτα τρία μεταφέρουν τη χρωματική πληροφορία για κάθε pixel, ενώ τα άλλα δύο σηματοδοτούν την έναρξη κάθε γραμμής σάρωσης (HSYNC) και κάθε πλαισίου (VSYNC). Στον υπολογιστή μας, τα δύο τελευταία παράγονται από τη GPU και τα τρία πρώτα από τον DAC, που συνδέεται στην έξοδο της μνήμης οθόνης.

Κρυμμένη αντίσταση

Το δίκτυο αντιστάσεων που λειτουργεί σαν DAC για τα δεδομένα της video RAM, μοιάζει πολύ διαφορετικό από εκείνο που συναντήσαμε στο κύκλωμα ήχου. Βέβαια, όπως θα δούμε και στη συνέχεια του άρθρου, το κύκλωμα γραφικών διαθέτει έξι διαφορετικούς μετατροπείς των 2 bits, ενώ το κύκλωμα ήχου διαθέτει έναν μετατροπέα των 8 bits. Παρ' όλα αυτά, ακόμα κι αν εστιάσουμε σε έναν από τους έξι "δίμπιτους" μετατροπείς, η σύνδεση των αντιστάσεων μοιάζει να μην έχει κανένα νόημα. Αυτό συμβαίνει γιατί οι DAC του κυκλώματος γραφικών συμπληρώνονται από μια αντίσταση που δεν υπάρχει στο δικό μας κύκλωμα. Πρόκειται για τη λεγόμενη αντίσταση εισόδου (input impedance) του κυκλώματος της οθόνης, η οποία ανέρχεται στα 75Ω. Σημειώστε ότι αυτή η αντίσταση υπάρχει εις τριπλούν, αφού η οθόνη διαθέτει μία τέτοια σε κάθε είσοδο χρώματος. Λαμβάνοντας υπόψη την ύπαρξη αυτής της αντίστασης, το κύκλωμα καθενός από τους "δίμπιτους" DAC αποκτά τη μορφή της εικόνας 13. Εφαρμόζοντας το θεώρημα της υπέρθεσης μπορούμε να υπολογίσουμε εύκολα την επίδραση κάθε εισόδου στην τάση εξόδου και, ακολούθως, τη σχέση που συνδέει τη "συνολική" τάση εξόδου με τις δύο εισόδους. Η εξίσωση στην οποία καταλήγουμε έχει ως εξής:

$$V_{out} = 69.7 * V_{bit0} / 2069.7 + 72.3 * V_{bit1} / 1072.3$$

Από την παραπάνω σχέση προκύπτουν δύο συμπεράσματα. Αφενός, η είσοδος Vbit1 έχει σχεδόν τη διπλάσια επίδραση από την είσοδο Vbit0 κι αυτό είναι απολύτως επιθυμητό, αφού το bit1 έχει τη διπλάσια αξία από το bit0. Αφετέρου, η τάση εξόδου δεν ξεπερνά ποτέ τα 0,5V. Αυτό μοιάζει να αποτελεί πρόβλημα, αλλά η "σμίκρυνση" των τάσεων πραγματοποιείται επίτηδες. Βλέπετε, σύμφωνα με τις προδιαγραφές του σήματος VGA, η τάση για κάθε χρωματική συνιστώσα πρέπει να κυμαίνεται μεταξύ 0 και 0,7V. Πρόκειται για έναν παράξενο περιορισμό, αλλά δεν τον επιβάλαμε εμείς. Ποιος ξέρει τι προσπαθούσαν να λύσουν οι μηχανικοί που σχεδίασαν το πρότυπο.



Εικόνα 13
Καθένας από τους έξι "δίμπιτους" DAC που ενσωματώνει η "κάρτα γραφικών", ολοκληρώνεται σαν κύκλωμα με την αντίσταση εισόδου της οθόνης. Για την ανάλυση επιστρατεύουμε μόνο το θεώρημα της υπέρθεσης, αφού το κύκλωμα είναι αρκετά απλό και δεν χρειάζεται κάποια απλοποίηση.

Δίκαιη κατανομή

Προηγουμένως αναφέραμε ότι το κύκλωμα γραφικών του υπολογιστή μας αποθηκεύει 8 bits για κάθε pixel, ενώ λίγο αργότερα είπαμε ότι αυτά τα bits μοιράζονται στις τρεις χρωματικές συνιστώσες. Δεν σας φαίνεται παράξενο αυτό; Πώς μπορεί να γίνει αυτή η διαίρεση, χωρίς να αλλοιωθούν τα χρώματα;

Η προφανής λύση θα ήταν να χρησιμοποιήσουμε δύο bits για τη μία χρωματική συνιστώσα κι από τρία bits για καθεμία από τις άλλες. Σε όσα συστήματα γινόταν κάτι τέτοιο, το κόκκινο και το πράσινο έπαιρναν από 3 bits, ενώ το μπλε έπαιρνε το περίσσευμα (2 bits). Παραδόξως, αυτό το χρωματικό σύστημα (RRR-GGG-BB) έδινε μια σχετικά ομοιόμορφη παλέτα, καθώς το ανθρώπινο μάτι είναι λιγότερο ευαίσθητο στο μπλε. Εμείς πάντως ακολουθήσαμε έναν διαφορετικό δρόμο, αφού η σκέψη και μόνο αυτής της ανισόρροπης κατανομής μας ενοχλούσε αφάνταστα. Η λύση που υλοποιήσαμε βασίζεται στον τρόπο λειτουργίας των κυκλωμάτων που διέθεταν 4 bits χρώματος. Σε αυτά τα κυκλώματα, τα πρώτα τρία bits αντιστοιχούσαν στις τρεις χρωματικές συνιστώσες, ενώ το τέταρτο bit ρύθμιζε την ένταση και των τριών. Το συγκεκριμένο χρωματικό σύστημα ονομαζόταν RGBI και μας έδωσε την ιδέα ενός συστήματος RR-GG-BB-II. Έτσι, αποφασίσαμε να δώσουμε από δύο bits σε κάθε χρωματική συνιστώσα και να χρησιμοποιήσουμε τα υπόλοιπα δύο για τη ρύθμιση της έντασης ολόκληρης της τριάδας. Με αυτόν τον καταμερισμό έχουμε 4 διαβαθμίσεις για κάθε χρωματική συνιστώσα και 4 στάθμες έντασης (ή φωτεινότητας) για το χρώμα που περιγράφουν τα πρώτα έξι bits. Το συγκεκριμένο σύστημα επιτρέπει την αξιοποίηση και των οκτώ διαθέσιμων bits, χωρίς να εισάγει την παραμικρή αλλοίωση στα χρώματα — αντιληπτή ή όχι!

Πολλαπλές μετατροπές

Το χρωματικό σύστημα RR-GG-BB-II δημιουργεί μια ομοιόμορφη παλέτα, αλλά καταφέρνει να περιπλέξει ελαφρώς την κατασκευή του κυκλώματος. Βλέπετε, το σήμα VGA είναι αναλογικό κι αυτό σημαίνει ότι τα 8 bits χρώματος πρέπει με κάποιον τρόπο να μετατραπούν σε τρεις αναλογικές τάσεις, μία για κάθε χρωματική συνιστώσα. Η μετατροπή των πρώτων έξι bits αποτελεί απλή υπόθεση: Αρκούν τρεις διαφορετικοί DAC, καθένας από τους οποίους θα διαθέτει μια είσοδο των δύο bits. Οι έξοδοι αυτών των DAC θα παρέχουν το αναλογικό σήμα για το κόκκινο, το πράσινο και το μπλε. Τι πρέπει να γίνει όμως με τα υπόλοιπα δύο bits; Εφόσον περιγράφουν τη συνολική ένταση του τελικού χρώματος, τα συγκεκριμένα bits πρέπει να μετατραπούν σε μια αναλογική τάση κι αυτή να προστεθεί σε καθεμία από τις τάσεις των τριών συνιστωσών. Η πρόσθεση όμως πρέπει να γίνει κατά τέτοιον τρόπο, ώστε να μη βραχυκυκλωθούν οι έξοδοι των DAC. Αν συμβεί αυτό το βραχυκύκλωμα, είναι άγνωστο το πώς θα συνδυαστούν οι τιμές για κάθε χρωματική συνιστώσα. Με άλλα λόγια, το χρώμα των διαφόρων pixels θα είναι εντελώς απρόβλεπτο. Για να αποφύγουμε το πρόβλημα του βραχυκυκλώματος, τα δύο τελευταία bits μετατρέπονται από *τρεις ξεχωριστούς* DAC, οι οποίοι απομονώνονται μεταξύ τους με τη βοήθεια μερικών διόδων. Με την ευκαιρία, να σημειώσουμε ότι χρησιμοποιήσαμε απλές "διόδους σήματος", όπως είναι οι 1N4148, κι όχι διόδους ανόρθωσης. Η επιλογή έγινε με το σκεπτικό ότι οι δίοδοι σήματος είναι αρκετά πιο γρήγορες από τις υπόλοιπες και θα μπορούν να ανταποκρίνονται πιο σωστά στις ταχύτατες μεταβολές των δεδομένων για τα pixels.

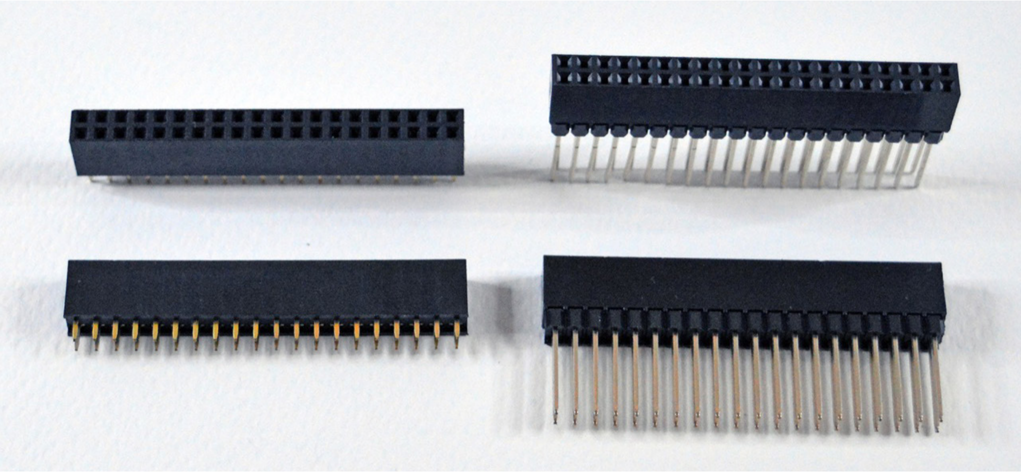
ΠΙΝΑΚΑΣ ΕΞΑΡΤΗΜΑΤΩΝ

ΕΞΑΡΤΗΜΑ	ΧΡΗΣΙΜΟΤΗΤΑ	ΚΟΣΤΟΣ
2 x ATMEGA644	CPU και GPU	12.0 €
1 x ATMEGA328	APU	2.0 €
1 x TC551001BPL-70L	μνήμη οθόνης	5.0 €
1 x 74HC541	απομόνωση data bus από DAC	1.0 € (για 2 τμχ.)
1 x 74HC573	διαχωρισμός data bus	1.0 € (για 2 τμχ.)
1 x 74LS04	αντιστροφή σημάτων	1.0 € (για 2 τμχ.)
3 x 20K resistor	pull-up για RESET μικροελεγκτών	1.0 € (για 100 τμχ.)
13 x 2K resistor	DAC για εικόνα και ήχο	1.0 € (για 100 τμχ.)
11 x 1K resistor	DAC για εικόνα και ήχο	1.0 € (για 100 τμχ.)
6 x 1N4148 diode	DAC για εικόνα	1.0 € (για 100 τμχ.)
2 x 22pF ceramic	πυκνωτές για κρύσταλο	1.0 € (για 100 τμχ.)
8 x 100nF ceramic	εξομάλυνση τροφοδοσίας ολοκληρωμένων	1.0 € (για 100 τμχ.)
3 x 100nF polyester	εξομάλυνση τροφοδοσίας πλακετών	2.5 € (για 20 τμχ.)
3 x 6.8μF tantalum	εξομάλυνση τροφοδοσίας μικροελεγκτών	2.0 € (για 10 τμχ.)
2 x 10uF electrolytic	μετάδοση σήματος ήχου	1.0 € (για 20 τμχ.)
1 x passive buzzer	εκκίνηση και σφάλμα πληκτρολογίου	1.0 € (για 5 τμχ.)
1 x LED 5mm	εκκίνηση και σφάλμα πληκτρολογίου	2.0 € (για 100 τμχ.)
1 x tactile switch	διακοπή προγράμματος	1.5 € (για 100 τμχ.)
1 x PS2 socket	σύνδεση πληκτρολογίου	1.5 € (για 3 τμχ.)
1 x 3mm audio socket	σύνδεση ηχείων	1.0 € (για 5 τμχ.)
1 x female VGA socket	σύνδεση οθόνης	1.0 €
2 x 20MHz oscillator	χρονισμός CPU και GPU	5.0 € (για 4 τμχ.)
1 x 16MHz crystall	χρονισμός APU	1.0 € (για 10 τμχ.)
3 x 9*15cm fiberglass prototyping board		11.0 €
6 x 2*20 female (long-pin) stackable pin header		16.0 € (για 20 τμχ.)
1 x 5V / 3A switching power supply		6.0 €

Πολλά από τα εξαρτήματα του υπολογιστή υπήρχαν ήδη στα συρτάρια μας, ενώ κάποια άλλα τα αγοράσουμε κατά δεκάδες, αφού αυτή η ποσότητα συνέφερε πολύ περισσότερο. Έτσι καταφέραμε να κατασκευάσουμε τον υπολογιστή και ταυτόχρονα να ανανεώσουμε το απόθεμα των εξαρτημάτων μας. Τελικά, το κόστος για τον υπολογιστή και μόνο ήταν πολύ χαμηλότερο από τα 81€ που υπονοεί ο πίνακας.

Στην πράξη

Ο διαχωρισμός του υπολογιστή σε τρία ξεχωριστά συστήματα δεν έγινε μόνο για θεωρητικούς λόγους, αλλά για να διευκολύνει τη σχεδίαση. Από την αρχή κιόλας του εγχειρήματος μας προβλημάτιζε η υλοποίηση και η (ισχυρή) πιθανότητα του να γίνουν λάθη. Πώς θα εντοπίζαμε τα bugs σε ένα τόσο μεγάλο κύκλωμα, αν το υλοποιούσαμε πάνω σε μία μόνο πλακέτα; Όπως υποψιάζεστε, όταν αποφασίσαμε να "σπάσουμε" τον υπολογιστή σε τρία τμήματα, αποφασίσαμε ταυτόχρονα ότι το καθένα θα βρισκόταν στη δική του πλακέτα. Φυσικά, οι εν λόγω πλακέτες θα έπρεπε κάπως να κουμπώνουν κι αυτός είναι ο λόγος που σε όλα τα κυκλώματα εμφανίζεται ένα ζεύγος από pin headers. Τα συγκεκριμένα εξαρτήματα φαίνονται στη δεξιά πλευρά της εικόνας 14 κι όπως μπορείτε να δείτε έχουν επιμήκεις ακροδέκτες: είναι τύπου stackable. Έτσι, οι πλακέτες που συνθέτουν τον υπολογιστή μπορούν να κουμπώνουν η μία πάνω στην άλλη, θυμίζοντας ένα πολυώροφο κτίριο. Οι ακροδέκτες που φαίνονται στα αριστερά της ίδιας φωτογραφίας χρησιμοποιήθηκαν μόνο σε μία πλακέτα, η οποία θα έπαιζε το ρόλο του "ισογείου". Η συγκεκριμένα πλακέτα θέλαμε να πατάει σε ειδικά πόδια και όχι πάνω στους ακροδέκτες της.



Εικόνα 14
Τα pin headers που φαίνονται στα δεξιά έχουν επιμήκεις ακροδέκτες και μπορούν να κουμπώνουν ο ένας πάνω στον άλλο, ακόμα κι όταν έχουν προσαρτηθεί σε πλακέτες. Ακριβώς γι' αυτό ονομάζονται και stackable.

Όπως θα έχετε προσέξει, δεν κατασκευάσαμε τυπωμένα κυκλώματα. Το όλο εγχείρημα ξεκίνησε σαν ένα πείραμα και δεν γνωρίζαμε εξ' αρχής αν θα ολοκληρωνόταν με επιτυχία ή όχι. Έτσι, αποφασίσαμε να υλοποιήσουμε τον υπολογιστή σε διάτρητες πλακέτες (prototyping boards), πραγματοποιώντας τις συνδέσεις με καλώδια. Παρεμπιπτόντως, το καλώδιο που χρησιμοποιήσαμε είναι μονόκλωνο και ο αγωγός του έχει διάμετρο 0.6mm (22AWG). Εσείς θα μπορούσατε να χρησιμοποιήσετε και καλώδιο διαφορετικής διαμέτρου, αλλά αυτό αποδείχτηκε αρκετά βολικό και σταθερό.

Στη μέχρι τώρα μελέτη είναι πιθανό να παρατηρήσατε τη χρήση αρκετών πυκνωτών, οι οποίοι συνδέονται στους ακροδέκτες της τροφοδοσίας. Οι κεραμικοί πυκνωτές των 100nF προορίζονται για την εξομάλυνση της τροφοδοσίας των ολοκληρωμένων κυκλωμάτων, ενώ ειδικά για τους μικροελεγκτές έχουμε επιστρατεύσει και από έναν πυκνωτή τανταλίου των 6.8μF. Εδώ υπάρχουν δύο κρίσιμες λεπτομέρειες. Αφενός οι πυκνωτές τανταλίου έχουν συγκεκριμένη πολικότητα (όπως και οι ηλεκτρολυτικοί), γεγονός που σημαίνει ότι η τοποθέτησή τους απαιτεί προσοχή. Αφετέρου όλοι οι πυκνωτές που αναφέραμε πρέπει να τοποθετηθούν όσο πιο κοντά γίνεται στους ακροδέκτες τροφοδοσίας των ολοκληρωμένων. Στις φωτογραφίες των κυκλωμάτων θα δείτε ότι τους έχουμε τοποθετήσει *ακριβώς* δίπλα στα ολοκληρωμένα, ενώ σε ορισμένες περιπτώσεις δεν θα τους δείτε καθόλου, αφού τους τοποθετήσαμε ανάμεσα στην πλακέτα και στο εκάστοτε τσιπ. Οι συγκεκριμένοι πυκνωτές έχουν σαν αποστολή την "απορρόφηση" των ρευμάτων που προκύπτουν από τις απότομες μεταβολές στα διάφορα ψηφιακά σήματα. Αυτά τα ρεύματα ονομάζονται transient currents και μολονότι έχουν σύντομο χρόνο ζωής, έχουν σχετικά μεγάλη ένταση. Ένα καλώδιο που διαρρέεται από τέτοια ρεύματα μπορεί εύκολα να λειτουργήσει σαν κεραία κι αυτός είναι ο λόγος που οι πυκνωτές πρέπει να βρίσκονται όσο πιο κοντά γίνεται στους ακροδέκτες τροφοδοσίας των ολοκληρωμένων. Εξετάζοντας τις φωτογραφίες θα διαπιστώσετε ότι έχουμε επιστρατεύσει και από έναν πυκνωτή πολυεστέρα (τους λεγόμενους "πλαστικούς"), για την εξομάλυνση της τροφοδοσίας κάθε πλακέτας. Οι συγκεκριμένοι πυκνωτές έχουν τοποθετηθεί δίπλα στους ακροδέκτες του pin header, οι οποίοι μεταφέρουν την τάση τροφοδοσίας του συστήματος. Πάντως, η αλήθεια είναι ότι η ύπαρξη αυτών των πυκνωτών αποτελεί υπερβολή.

Εν είδει στέγης για τις πλακέτες που απαρτίζουν τον υπολογιστή χρησιμοποιήσαμε μια ακόμα πλακέτα, των ίδιων διαστάσεων. Για αυτήν δεν σχεδιάσαμε κάποιο κύκλωμα, αφού επάνω της κουμπώσαμε ένα τροφοδοτικό των 5V / 3A και συνδέσαμε σε σειρά έναν διακόπτη. Το μόνο που χρειάστηκε να προσέξουμε σε αυτό τον "όροφο" του υπολογιστή ήταν να συνδέσουμε την τροφοδοσία στους σωστούς ακροδέκτες του pin header και φυσικά με τη σωστή πολικότητα.

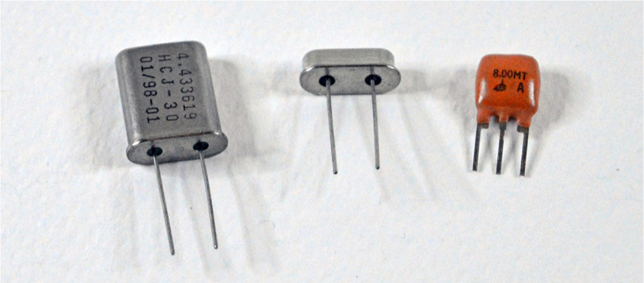
Σε αυτό το σημείο πιστεύουμε ότι μπορούμε να σταματήσουμε την ακατάσχετη πολυλογία. Πλέον, γνωρίζετε όλα τα μυστικά του hardware του υπολογιστή μας και μπορείτε να τον αναπαράγετε. Μα, γιατί κάθεστε; Δεν θα βάλετε το κολλητήρι στην πρίζα;

Κρύσταλλοι και κεραμικά

Η τεχνική ονομασία για τα κυκλώματα που παράγουν τους παλμούς συγχρονισμού είναι "ταλαντωτές" (oscillators). Αυτά τα κυκλώματα περιλαμβάνουν έναν ενισχυτή σήματος, του οποίου η έξοδος ανατροφοδοτείται στην είσοδο. Προσέξτε τώρα το εξής: Σε αυτόν το βρόχο ανάδρασης βρίσκεται ένα φίλτρο που επιτρέπει τη διέλευση ηλεκτρικών ταλαντώσεων μιας συγκεκριμένης συχνότητας. Όταν ενεργοποιείται το κύκλωμα ο ενισχυτής δέχεται στην είσοδό του τις τυχαίες ηλεκτρικές ταλαντώσεις που προκαλούνται από το θερμικό θόρυβο και τις

ενισχύει. Από όλες αυτές τις ταλαντώσεις, στην είσοδο του ενισχυτή επιστρέφει μόνο εκείνη που επέτρεψε το φίλτρο κι έτσι ενισχύεται ξανά! Με αυτό το μηχανισμό, μέσα σε ελάχιστο χρονικό διάστημα το κύκλωμα καταλήγει να παράγει μια ταλάντωση σταθερής συχνότητας και μεγάλου πλάτους.

Οι περισσότεροι μικροελεγκτές ενσωματώνουν όλα τα τμήματα ενός ταλαντωτή, εκτός από το φίλτρο. Αναρωτιέστε για ποιο πράγμα μιλάμε; Μα, φυσικά, για τους κρυστάλλους και τα αντίστοιχα κεραμικά. Τα εν λόγω εξαρτήματα παίζουν το ρόλο του φίλτρου, στο βρόχο ανάδρασης ενός ενισχυτή. Για παράδειγμα, ένας κρύσταλλος των 16MHz λειτουργεί σαν φίλτρο που επιτρέπει τη διέλευση μόνο των ηλεκτρικών ταλαντώσεων με συχνότητα 16MHz. Το πώς επιτυγχάνεται αυτό το φιλτράρισμα σχετίζεται με ένα φαινόμενο που ονομάζεται πιεζοηλεκτρικό, καθώς και με τις διαστάσεις του κρυστάλλου. Οι κρύσταλλοι θεωρούνται πολύ αξιόπιστοι για την παραγωγή παλμών σταθερής συχνότητας. Κάτι παρόμοιο ισχύει και για τα αντίστοιχα κεραμικά εξαρτήματα, αν κι αυτά παρουσιάζουν μεγαλύτερη ευαισθησία στις μεταβολές της θερμοκρασίας. Πάντως, για τον υπολογιστή μας,προτιμήσαμε να στηριχτούμε σε μια ακόμα πιο αξιόπιστη λύση: Τους ολοκληρωμένους κρυσταλλικούς ταλαντωτές που, εκτός από τον ενισχυτή και τον κατάλληλο κρύσταλλο, ενσωματώνουν σύστημα αντιστάθμισης των μεταβολών της θερμοκρασίας.



Εικόνα 15
Στην εικόνα βλέπουμε δύο κρυστάλλους και ένα κεραμικό στοιχείο, που χρησιμοποιούνται ως φίλτρα στους βρόχους ανάδρασης των ταλαντωτών. Οι κρύσταλλοι παρουσιάζουν μεγαλύτερη σταθερότητα, αλλά η χρήση τους απαιτεί και την παρουσία δύο πυκνωτών. Στα αντίστοιχα κεραμικά εξαρτήματα οι πυκνωτές είναι ενσωματωμένοι.



Εικόνα 16
Η συσκευασία αυτών των εξαρτημάτων θυμίζει τους κρυστάλλους. Πρόκειται για ολοκληρωμένους ταλαντωτές που, εκτός από τον κρύσταλλο, περιλαμβάνουν τον απαιτούμενο ενισχυτή και ένα κύκλωμα αντιστάθμισης των μεταβολών της θερμοκρασίας.

COOKIES


STEPHEN HAWKING...ΤΙ ΝΑ ΠΡΩΤΟΠΕΙ ΚΑΝΕΙΣ...
...ΚΑΠΟΪΟΙ ΤΟΝ ΘΕΩΡΟΥΝ ΤΟΝ ΤΙΤΑΝΑ ΤΗΣ ΕΠΙΣΤΗΜΗΣ...
...ΓΙΑ ΑΛΛΟΥΣ ΕΙΝΑΙ ΤΟ ΙΕΡΟ ΤΟΤΕΜ ΤΗΣ ΚΟΣΜΟΛΟΓΙΑΣ...
...Ο ΓΙΓΑΣ ΤΗΣ ΘΕΩΡΗΤΙΚΗΣ ΦΥΣΙΚΗΣ...
...Ο ΑΠΟΛΥΤΟΣ ΦΑΡΟΣ ΤΟΥ ΟΡΘΟΥ ΛΟΓΟΥ...
ΠΑΝΟΣ
...Ο ΛΕΜΠΡΟΝ ΤΖΕΗΣ ΤΗΣ ΣΚΕΨΗΣ!!!
ΠΕΣ ΚΑΤΙ, ΔΑΣΚΑΛΕ!
ΕΑΝ ΟΙ ΜΗΧΑΝΕΣ ΠΑΡΑΓΟΥΝ ΟΛΑ ΟΣΑ ΧΡΕΙΑΖΟΜΑΣΤΕ, ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΕΞΑΡΤΑΤΑΙ ΑΠΟ ΤΟ ΠΩΣ ΔΙΑΝΕΜΟΝΤΑΙ. ΝΑ ΦΟΒΑΣΤΕ ΤΟΝ ΚΑΠΙΤΑΛΙΣΜΟ, ΟΧΙ ΤΙΣ ΜΗΧΑΝΕΣ...
...ΕΝΩ ΣΤΗΝ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ ΕΙΝΑΙ ΕΝΑΣ ΥΠΕΡΤΙΜΗΜΕΝΟΣ ΠΑΠΠΟΥΣ, ΠΟΥ ΑΝ ΗΤΑΝ ΥΓΙΗΣ ΔΕΝ ΘΑ ΤΟΝ ΗΞΕΡΕ ΟΥΤΕ Η ΜΑΝΑ ΤΟΥ!